1980

A dimensionality reduction technique for enhancing information context.

Maurer, Michael Lee

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/18953

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A DIMENSIONALITY REDUCTION TECHNIQUE
FOR ENHANCING INFORMATION CONTEXT

by

Michael Lee Maurer

June 1980

Thesis Advisor: L. A. Wilson

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Dimensionality Reduction Technique For Enhancing Information Context		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis;June1980
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael Lee Maurer		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey,California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey,California 93940		12. REPORT DATE June 1980
		13. NUMBER OF PAGES 184
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey,California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

data representation, dimension reduction,dimensionality reduction, feature extraction,feature extractor,feature selection, feature transformation,information context,information representation, pattern recognition,pattern classification,radar ship classification radar target identification,relative proximity of information

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A computer processing technique is advanced which seeks to retai or improve data information context while reducing the dimensionalit of data representation. Defining information context as the relative proximity of data points, a nonlinear transformation is analytically derived which utilizes Euclidean distance to one or more reference points to provide a measure of similarity between data points. The nonarbitrary reference points are selectively manipulated to provide,

given certain constraints, a unique mapping from high dimensional space to one or more dimensions for each point in space. The transformation process enhances class clustering and interclass separation in the lower dimensional representation.

Computer processed experimental results are presented of reduction from 32, 10, and 3 space into 2 space for both synthetic and real world data. Utilizing a ratio of intraclass variance to interclass variance as a figure of merit and as one possible optimization criterion, this technique yeilded a significant ratio improvement in mapping from higher dimensional space into 2 dimensional space for all cases examined.

A DIMENSIONALITY REDUCTION TECHNIQUE FOR ENHANCING INFORMATION CONTEXT

BY

Michael Lee Maurer
Lieutenant, United States Navy
B.S., University of Texas at Arlington, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June, 1980

ABSTRACT

A computer processing technique is advanced which seeks to retain or improve data information context while reducing the dimensionality of data representation. Defining information context as the relative proximity of data points, a nonlinear transformation is analytically derived which utilizes Euclidean distance to one or more reference points to provide a measure of similarity between data points. The nonarbitrary reference points are selectively manipulated to provide, given certain contraints, a unique mapping from high dimensional space to one or more dimensions for each point in space. The transformation process enhances class clustering and interclass separation in the lower dimensional representation.

Computer processed experimental results are presented of reduction from 32, 10, and 3 space into 2 space for both synthetic and real world data. Utilizing a ratio of intraclass variance to interclass variance as a figure of merit and as one possible optimization criterion, this technique yielded a significant ratio improvement in mapping from higher dimensional space into 2 dimensional space for all cases examined.

TABLE OF CONTENTS

TABLE OF SYMBOLS

$X_i = (x_{1i}, x_{2i}, \ldots, x_{ni})$	a vector of length n
x_{ij}	an element of a vector
R_i	the i th reference point vector
p_{ij}	the j th element of R_i
d_l	the l th element of a vector which is the distance to the i reference point
T	a nonlinear transformation
k	a constant scaling factor
a_i	the maximum value the i th element od a vector may assume
M_i	mean vector for class i
w_i	number of members of class i
S_W	scatter within classes
$(X)^T$	transpose of the X vector
c	number of classes
S_B	between classes scatter
M_T	total class mean
z	total number of members of all classes
I_q	information context ratio for q dimension classes
$\neg$	not
$\wedge$	and

ACKNOWLEDGEMENTS

The author wishes to express his appreciation for the support and guidance of Dr. Wilson. The interaction with Dr. Hamming has been both thought-provoking and stimulating. For that I am grateful.

A special and sincere thanks goes to my wife, Aniece, for her support and sacrifice.

I. INTRODUCTION

> "In the widest sense, patterns are the means by which we interpret the world."
>
> WILLIAM S. MIESEL

Data becomes meaningful information when it provides a perspective on known information. In its most general sense, the context of unknown information is judged by its relationship to known information. The most numerous pattern recognition mechanism on the planet Earth, man, primarily judges the meaning of new information on his past experiences. He interprets his senses relative to his personal environment and experiences. However, the methods by which man preceives new experiences relative to past experiences is not yet fully understood.

Mathematical pattern recognition, using digital computers, attempts to emulate the human's skill at pattern recognition, albeit poorly, by relating new information to its own data base of accumulated information. The computer has the advantage of performing pattern recognition in high dimensional spaces, spaces incomprehensible to man.

A. MOTIVATION

Pattern recognition represents information as numerical

values. In supervised learning, numerical representations of known objects are compared against numerical representations of unknown objects in an attempt to recognize the unknown object. The complexity of analysis rapidly increases as more and more measures of an object are collected. Each measure is, in numerical form, a descriptor of some attribute of an item, be it a physical object, an event in time, or some metaphysical relationship. The number of measures of a item of data define the number of dimensions in space in which the item exists. A sample of an object is defined to be one set of measurements of that object. The complexity of evaluating the meaning or identity of an sample increases exponentially as the number of dimensions in which the object is described. The fact that procedures which are analytically or computationally manageable in low dimensions become completely impractical with high dimensional representations is termed within the pattern recognition literature as the "curse of dimensionality"[1,2,3].

The significance of information is not in its representation but in its context. In classifying data relative to known information the concern is recognition, not representation. The premise of this research is that a transformation exist which will overcome, to some extent, the curse of dimensionality by reducing information representation while retaining context. Viewed geometrically, this transformation will attempt to retain or

enhance relative proximity of similiar information and separate the relative proximity between different information sources while reducing the number of dimensions in the representation.

B. A RELATIVE PROXIMITY SCENARIO

Consider the problem of an aircraft navigator equipped with a range only measuring device. The navigator knows the general location of his aircraft but would like to precisely fix his position. In doing so, he measures the distance to two landmarks conveniently available to him as in figure 1.1. He then circumscribes a cirle around each of the landmarks ,each with a radius equal to the distance from the aircraft to that landmark. Unless he is exactly on the line drawn between the two references points the circles drawn will intersect at two points. By knowing his general position, that is, by knowing the aircraft's position relative to that line, the nagivator can resolve the ambiguity and select the correct intersection as his position.

This example, greatly generalized and viewed from the perspective of the landmarks rather than the navigator, is the transformation developed in this thesis to reduce representation while retaining relative proximity of similiar information.

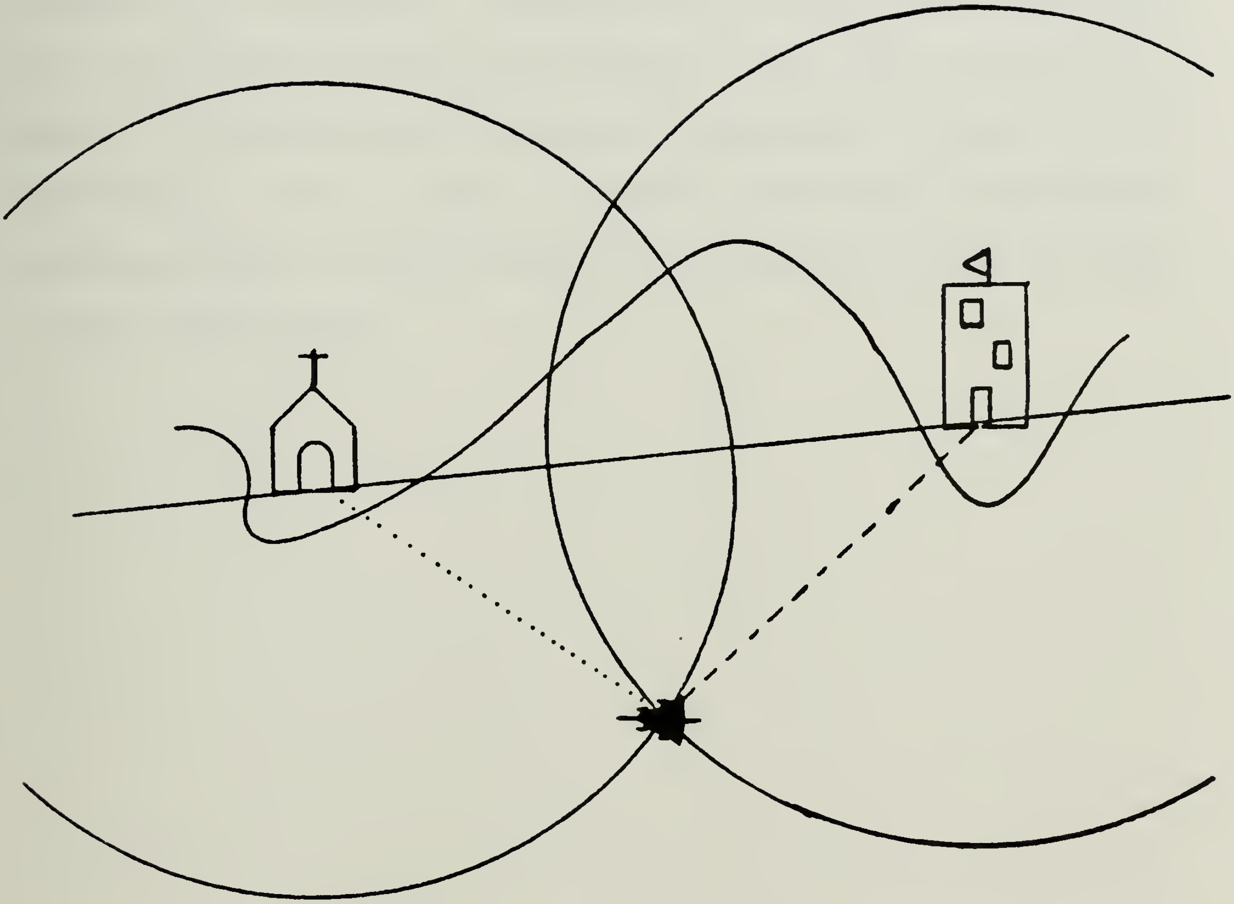


Figure 1.1 Aircraft position determined relative to two reference points.

C. SCOPE

The generalized pattern recognition process is described to provide insight into the role of dimensionality reduction. Next the nonlinear transformation employed to reduce data representation is analytically derived. This is followed by geometric illustrations of the Transformation and a rationalization of its effects on class clustering. Results of specific test cases are described and graphically illustrated. The final chapter provides conclusions, recommendations for applications, and further research areas in this methodology.

II. THE PATTERN RECOGNITION PROCESS

A. NATURE OF THE PROCESS

"Pattern classification is the assignment of a physical object or event to one of several prespecified catagories "[1]. The act of making that assignment can be characterized by three sequential logical component processes as shown in figure 2.1. In the first process, the physical world is sensed by some transducer system which transforms data into a machine processible state. The transducer changes the physical reality of an object, characterized as a continuum of parameters and infinite in dimensionality, into a pattern space whose domain is defined by the discretization of sensor data observed in the real world. This discrete set of measurements finitely bounds the range of values and number of dimensions which characterize the object. Feature space is an intermediate domain between pattern space and classification space. There may be one or more subprocesses required in transforming pattern space into feature space. This transformation into feature space is the process, termed feature selection, preprocessing, or feature extraction, by which a sample representation in pattern space is described by a finite and usually smaller sample representation called features. Feature space is a reduced

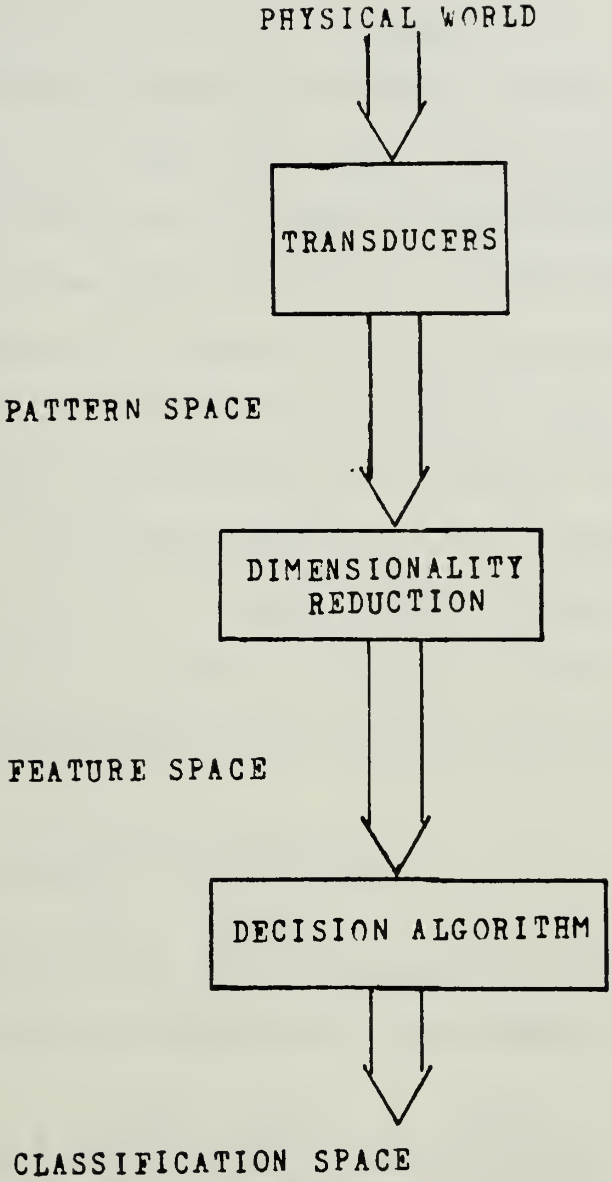


FIGURE 2.1 The pattern recognition process

representation which attempts to retain as much discriminating power as possible while removing as much redundancy as possible. The transformation from feature space to classification space is accomplished via a set of decision rules which classify an unlabeled (unknown) sample of an object as a member of one of the known data sets. The classification problem is basically one of partitioning the feature space into regions, one region for each category. The view of pattern recognition as a series of processes was provided by H. C. Andrews[3]. Meisel [2] views pattern recognition as a series of states in which the data exist. The two perspectives are logically equivalent.

B. DIMENSIONALITY REDUCTION

In describing the dimensionality reduction inherent in feature extraction Duda and Hart comment :

> There is a growing body of theory of dimensionality reduction for pattern classification. Some of these methods seek to form new features out of linear combinations of old ones. Others seek merely a small subset of the original features. A major problem confronting this theory is that the division of pattern recognition into feature extraction followed by classification is theoretically artificial. A completely optimal feature extractor can never be anything but an optimal classifier.[1]

Andrews states :

> While the objective in defining the feature space is to reduce the dimensionality of the pattern space yet

> maintaining discriminatory power for classifiction purposes, successful transformations still seem in their infancy. There exist a variety of linear transformations as well as some nonlinear methods which are developing particular appeal but the real frontiers of pattern recognition still lie ahead in developing a viable feature selection transformation that undoes the redundant data gathering inherent in the definition of pattern space.[3]

The variety of feature extraction techniques are too numerous to mention individually, but can be discussed as families of methodologies. Principal component analysis techniques attempt to maintain discrimination while reducing the dimensionality of data representation by selecting a subset of measurements from pattern space which contain the most variability. The objective of factor analysis is to find a lower dimensional representation that accounts for the correlations among the features. The multidimensional scaling technique reduces the dimensionality while attempting to maintain the distance relationships between all points in pattern space in feature space. This feature extractor iteratively processes the data until a minimum error exists in the feature space representation of pattern space distance relationships. Classical discriminant analysis attempts to find a lower dimensional surface on which to project the data samples and achieve good separation between classes. In each case the methodologies incur some loss of information in feature extraction.

The intrinsic meaning of the data points, not their

representation, is their single most important property. An optimal feature extractor must not lose context while reducing data representation. Certainily, any feature extractor which reduces the context while reducing the representation is suboptimal. Yet the term feature extractor itself implies retaining some information while discarding some information. Perhaps a change of perspective is required to better reduce data representations. Consider a method of representing in a lower dimension the relationships between data samples in pattern space rather than the data itself. The relationships are the significant factors, not the representations, for they define the context of the information.

The relationships between data points are typically judged in terms of distance measures. Meisel astutely noted "distance is crucial in pattern recognition; it is assumed, roughly, that the closer a point is to another point, the more similar are the patterns represented by those points".[2] Multidimensional scaling capitalizes on this fact by attempting to retain pattern space distance relationships in feature space. Yet it fails to do so completely since the distance relationships between all points can not be explicitly retained in any less than the number of pattern space dimensions unless the points exist in a subspace within the pattern space.

Again recall that the significant factor is context, not

representation. Taking the liberty of paraphasing Meisel's words, the greater the relative proximity of one point to another point, the greater the similarity of the patterns represented by those points. If, in reducing the data representation dimensionality, the relative proximity of similar information is maintained then it seems intiutive that little loss of context has occured in the transformation. Multidimensional scaling severly constraints itself by attempting to maintain distance relationships in lower dimensions. The transformation presented here will forego maintaining distance relationships between all points in lieu of maintaining relative proximity between similar data points.

C. GENERAL APPROACH TO A RELATIVE PROXIMITY TRANSFORMATION

In developing this nonlinear transformation use is made of two axioms :

1. the distance between two points in n dimensional space is a scalar value;
2. an n dimensional lattice space is relatively sparse compared to n dimensional continuous space.

These facts are of vital signifinance in the ability to reduce representation without loss of context.

The distance from one point to another point is a

measure of the relative proximity of the points. In one dimension, the similarity or lack of similarity in distance from a reference point to all other points in effect defines each point's relative proximity to one another. By generalizing to n dimensions and constraining the data space to remove ambiguity, the context of points in n dimensions may be measured by their similarity in distance to one or more known reference points.

The selection of reference points will be based on a criterion of retaining, if not improving, relative proximity of similar representations, separating dissimilar representations, and providing a unique mapping from n dimensional pattern space to an m (m > 1) dimensional feature space.

III. N-DIMENSIONAL NONLINEAR TRANSFORMATIONS

This chapter considers the representation of known data samples in pattern space. Their representations in pattern space produce several constraints and assumptions about that space which allow nonlinear transformations to reduce the dimensionality of their representations. In subsequent chapters it will be shown that the reduced representation retains or, more likely, improves any clustering present in samples of the same object. This implies that the information context of the reduced representation is at least maintained if not improved in the lower dimension representation. The case of a two dimensional pattern space to two dimensional distance space transform is first developed, followed by a three dimensional pattern space to two dimensional distance space transformation. A discussion of n dimensional to m dimensional transforms conclude the chapter.

A. DEFINITIONS

Let $X_i = (x_{1i}, x_{2i}, \ldots, x_{ni})$ be the i th sample vector in pattern space describing an object where :

x_j $(j = 1,n)$ is a real number;

n is the finite number of dimensions in

the sample vector.

Let $D_i = (d_{1i}, d_{2i}, \ldots, d_{mi})$ be the ordered representation of the i th sample in distance space[1] where :

m	is the finite number of dimensions in the distance vector.
d_k $(k = 1,m)$	is a real number where (d_k) is the Euclidean distance from a designated reference point (R_k) to a sample point (X_i);

The Euclidean distance (d_k) is defined to have the following properties for any three distinct points (x_i, x_j, x_k) $i \neq j$:

$$d(x_i,x_j) = [\sum_{h=1}^{n}(x_{hi} - x_{hj})^2]^{1/2} \qquad d(x_i,x_j) > 0$$

$$d(x_i,x_i) = 0 \qquad d(x_i,x_j) = d(x_j,x_i)$$

$$d(x_i,x_j) + d(x_j,x_k) \geq d(x_i,x_k)$$

A nonlinear transformation (T) is defined such that

$$T : (X_i \dashrightarrow D_i)$$

B. TRANSFORMATION FROM TWO DIMENSIONAL PATTERN SPACE INTO TWO DIMENSIONAL DISTANCE SPACE

In considering the two dimensional transformation (T) recall the example of the navigator. In that situation, the navigator knew the aircraft's approximate position,

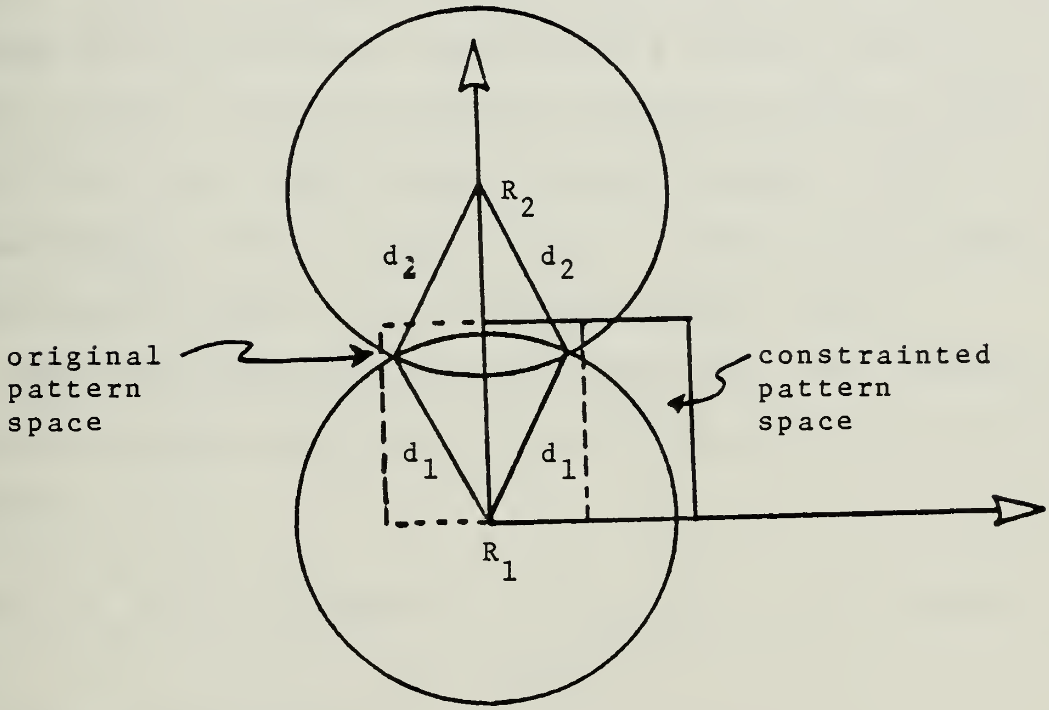
---

[1] The term distance space is utilized in lieu of feature space since the features are distances from reference points rather than a subset of measurements from pattern space as might be done , for example, in principal component anaylsis.

effectively limiting the area in which he could be precisely located. In the general case in two dimensions one method to locate a point in space precisely is to constrain that point to a certain region as in the case of the navigator. In constraining a point to remove ambiguity two factors enter into consideration. In supervised learning an attempt is made to identify an object solely on its proximity to other identified points. The range of values of the known objects are precisely defined and the unknown sample values will be approximately equal to the known values. Secondly, recall that the navigator had to know his general position relative to a line drawn between the two reference points to resolve the ambiguity in choosing which of the two intersections was his position. Knowing the approximate range of values allows selection of a constant scaling factor which will scale all points into nonnegative space. By scaling the data to be nonnegative most ambiguity is removed as all intersections occuring outside nonnegative space can be rejected, as in figure 3.1a.

To remove any ambiguity caused by both intersections occuring within the constrainted pattern space restrictions are placed on reference points R_1 and R_2. For computational simplicity reference point one (R_1) is defined as the origin. A valid reference point two (R_2) is defined such that one and only one intersection will occur within pattern

Figure 3.1a A two dimensional representation of a valid reference point two.



This reference point two is valid for this situation in that only one point of intersection lies within constrainted pattern space.

space[2]. The values of valid reference point twos (R_2) are generally not bounded and are infinite in number even with the uniqueness constraint. The distance from each reference point to a point in the pattern space lattice defines the radius of a circle. The intersection of the circles about each of the reference points defines a point in space. Figure 3.1a illustrates the case of a valid R_1 and R_2. Note that of the two intersections defined by the circles only one exist in constrainted pattern space. Figure 3.1b demonstrates an invalid R_2 in that two points of intersection exist in constrainted pattern space.

The two dimensional pattern space into two dimensional distance space transformation may be stated in the following manner :

Given a set of samples $Y_i = (y_{1i}, y_{2i})$, (i = 1,number of samples)

in pattern space where

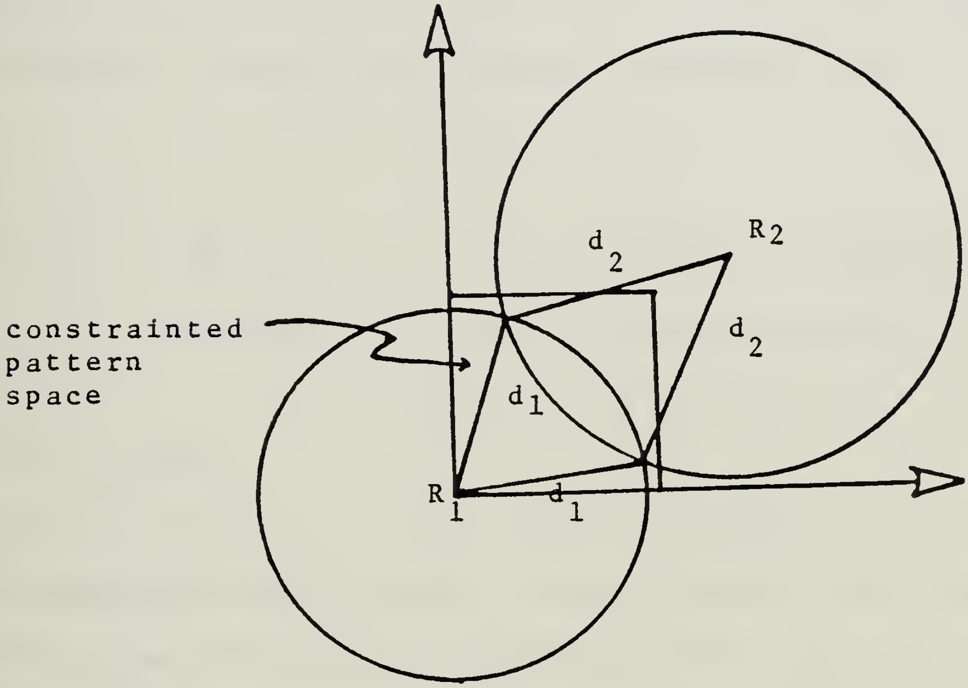
$0 < y_{ji} + k < a_i \quad (j = 1,2)$

a_i	is the maximum value which the i th element may assume;
k	is a scale factor such that the minimum value of $y_{ji} + k \geq 0$;

---

[2] From this point forward in the discussion, the term pattern space implies a nonnegative, maximally bounded pattern space.

Figure 3.1b A two dimensional representation of an invalid reference point two.



Ambiguity exists as there are two points of intersection defined within constrainted pattern space.

$x_{ji} = y_{ji} + k$

$X_i = (x_{1i}, x_{2i}), (i = 1, \text{number of samples})$ is the set of samples in constrainted pattern space

there exist a set of samples $D_i = (d_{1i}, d_{2i})$ in distance space obtained through the nonlinear transformation

$$T : (X_i \dashrightarrow D_i)$$

where

$d_{ji} = [\sum (x_{ji} - r_j)^2]^{1/2} (j = 1,2), (i = 1, \text{number of samples})$ is the Euclidean distance function as defined above {3.1}

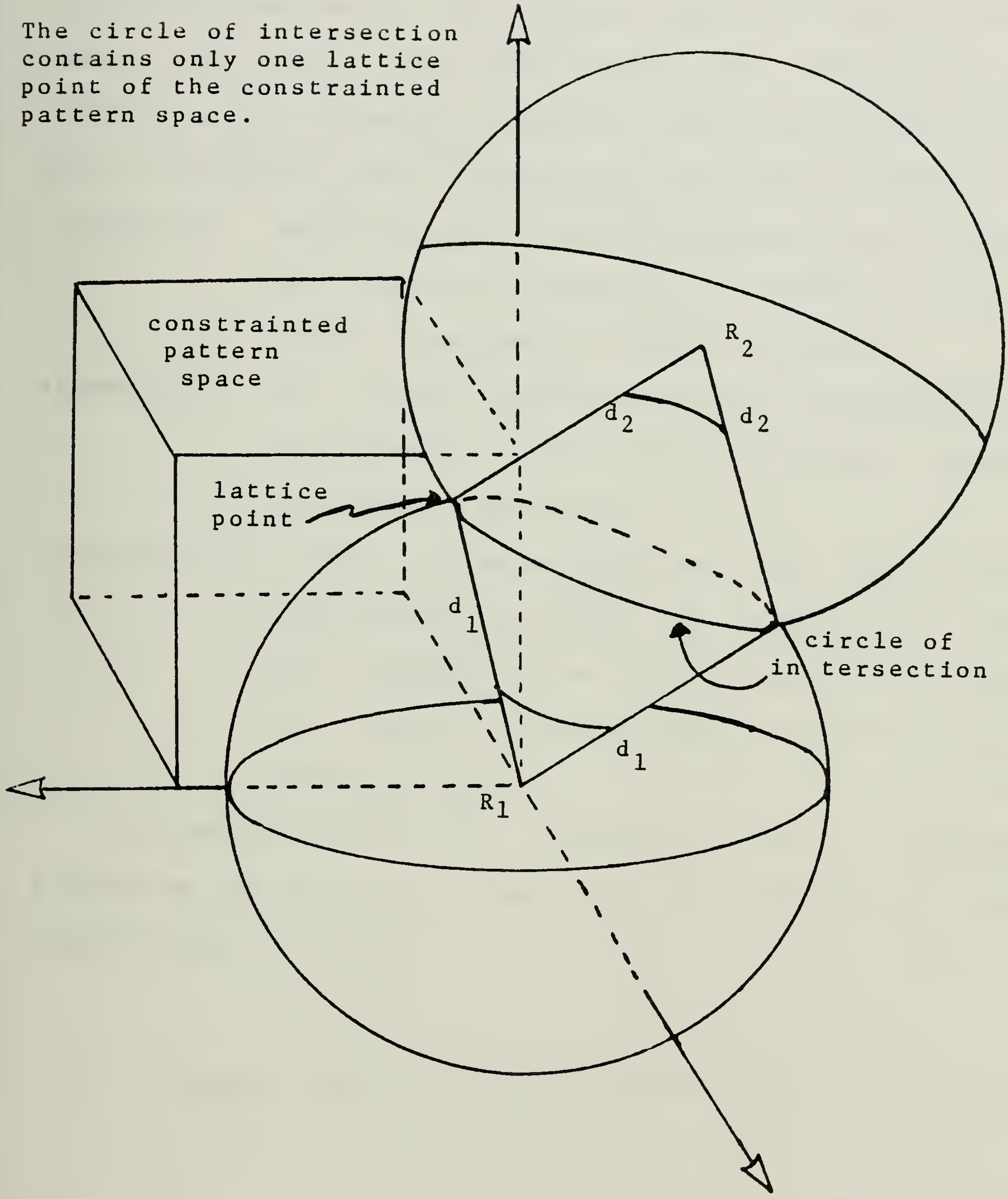
$R_1 = (0,0)$

$R_2 = (p_1, p_2)$ is a valid reference point.

A reference point two is valid for two space if and only if there exist one and only one solution within pattern space to the simultaneous equations defining (d_1) and (d_2) for all points in pattern space.

C. TRANSFORMATION FROM THREE DIMENSIONAL PATTERN SPACE INTO TWO DIMENSIONAL DISTANCE SPACE

Observe in figure 3.2 the radii (d_1, d_2) computed as the distance from a reference point to a sample point geometrically describe spheres in three dimensions. The intersection of the two spheres formed, respectively, of the radii d_1, d_2 from R_1, R_2 defines a circle in two dimensions.



The circle of intersection contains only one lattice point of the constrainted pattern space.

Figure 3.2 A three dimensional valid reference point two.

In continuous space all points on the circle of intersection are equally defined by the ordered pair (d_1, d_2) in two space. To remove this most uncomfortable ambiguity of an infinite number of points in pattern space mapping into one point in distance space consider the following assumptions :

assumption 1 pattern space may be represented as a lattice of discrete nonnegative, maximally bounded points separated by a unit distance;

assumption 2 the circle described by the intersection of the spheres is constructed to contain one and only one lattice point;

assumption 3 a valid reference point two (R_2) exist which manipulates the circle of intersection such that assumption two will be true for all lattice points in the pattern space defined by assumption one.

Assumption three is logically valid, provided assumption one is true, if and only if there exist the ordered pairs

$$D_i = (d_{1i}, d_{2i}) \text{ and } D_j = (d_{1j}, d_{2j}) \text{ for all } X_i,\ X_j \quad i \neq j$$

discrete sample points in three dimensional pattern space where

$$\neg [(d_{1i}=d_{1j}) \wedge (d_{2i}=d_{2j})] \text{ for all } D_i,D_j \quad i \neq j \qquad \{3.2\}$$

Restating equation 3.2 such that when the following equivalent equations are both false for all points X_i,X_j ($i \neq j$) in pattern space $R_2 = (p_1,p_2,p_3)$ is a valid reference point.

$$d_{1i}^2 - d_{1j}^2 = 0 \qquad \{3.3\}$$
$$d_{2i}^2 - d_{2j}^2 = 0 \qquad \{3.4\}$$

Expanding equation 3.4, in considering a three space into two space transformation, yields :

$$(X_i - R_2)^2 - (X_j - R_2)^2 = 0 \qquad \{3.5\}$$

Squaring and collecting terms results in :

$$x_{1i}^2 + x_{2i}^2 + x_{3i}^2 - x_{1j}^2 - x_{2j}^2 - x_{3j}^2 + 2x_{1j}p_1 + 2x_{2j}p_2 + 2x_{3j}p_3 - 2x_{1i}p_1 - 2x_{2i}p_2 - 2x_{3i}p_3 = 0 \qquad \{3.6\}$$

Equation 3.1 allows substitution of D_{1i}^2 for $(x_{1i}^2 + x_{2i}^2 + x_{3i}^2)$ and $-D_{1j}^2 = (-x_{1j}^2 - x_{2j}^2 - x_{3j}^2)$ and utilizing vector

notation for R_2, x_i^{2}'s and x_j^{2}' then

$$D_{1i}^{2} - D_{1j}^{2} + 2R_2 * (X_j - X_i) = 0 \qquad \{3.7\}$$

Equation 3.7 is the logical complement to equation 2.2 since only when the equality holds will a point $R_2 = (p_1, p_2, p_3)$ be invalid.

Assumption three, that a valid reference point two will always exist when assumption one is true, can be proven in the following manner. Two sets of constants are present in equation 3.7, the differences between the distance ones squared and the differences between the data points in pattern space for all points (X_i, X_j), $(i \neq j)$. Selecting the largest difference in magnitude[3] between distance ones squared and the minimum difference in magnitude between data points (X_i) and (X_j) then a reference point two can easily be selected such that the magnitude of the dot product of the candidate reference point two and the minimum difference between data points (X_i, X_j) is always greater than the maximum difference in magnitude between distance ones squared. Once the minimum reference point two is found then

---

[3] The minimum difference in magnitude is that difference whose absolute value is closest to the origin. Conversely, the maximum difference in magnitude is that difference whose absolute value is most distant from the origin.

every linear combination of that point is a valid reference point. Furthermore, every point larger in value for the maximum case and smaller in value for the negative case will be valid except for the case $p_i = p_j = p_k$ in symmetric pattern space.

Consider the example of a three dimensional pattern space which is exhaustively defined for all linear combinations of integer lattice points in the range of 0 - 6. The magnitude of the maximum $(D_{1i}^2 - D_{1j}^2)$ is 108. The minimum magnitude of the difference $(X_j - X_i)$ is the point (0,0,1). In accordance with equation 3.7, a reference point two is easily found such that

$$\text{max magnitude}(D_{1i}^2 - D_{1j}^2) < 2\,(R_2 * (\text{min magnitude}(X_j - X_i))) \qquad \{3.8\}$$

For this example,

$$108 < 2(\ R_2 * (\ 0,0,1))$$

implies $R_2 = (0,0,55)$

In effect, assumption three states there will always exist a unique mapping from three space to one space for discrete data.

When equation 3.3 is considered in addition to equation 3.4 then another degree of freedom is present. The net result of this combination is that valid reference point twos will exist inside the bounds defined by considering equation 3.4 alone. At present valid reference point twos

inside the bounds can only be determined by exhaustive searches attempting to validate a candidate reference point two with equation 3.7.

D. TRANSFORMATION FROM N DIMENSIONAL PATTERN SPACE INTO M DIMENSIONAL DISTANCE SPACE

Generalizing figure 3.2 into n dimensional space, the radius of intersection forms a (n - 1) dimensional hypersurface in n space. All the constraints and assumptions of the three space transform remain valid for the n dimensional case. The derivation in the previous section that a valid reference point two always exist in three space is but one case of the n dimensional argument. To extend the proof to n space simply increase the indices of the vectors to the desired value of n. There is no difficulty in mapping a (n - 1) dimensional hypersphere defined by the radius of intersection of m spheres into a single, unique point in m space where m ranges from one to infinity.

The n dimensional transformation may be stated as :

Given a set of sample data points $Y_i = (y_{1i}, y_{2i}, \ldots, y_{ni})$, (i = 1, number of samples) in pattern space where there exists a set of points $X_i = (x_{1i}, x_{2i}, \ldots, x_{ni})$, (i = 1,number of samples) where X_i is constrainted to be

1. nonnegative $y_{ji} + k \geq 0$

$x_{ji} = y_{ji} + k$

2. maximally bounded $x_{ji} \leq a_i$

3. a member of the set of lattice points with a unit separation between points $x_{ji} \in \{P\}$ $P = \{\text{all lattice points}\}$

k is a constant scale factor such that all possible points in data space are nonnegative values

a_i is the maximum value a (x_i) may assume

. there exist a set of sample points $D_i = (d_{1i}, d_{2i}, \ldots d_{mi})$ in distance space

where

$d_{bi} = [\sum (x_{ji} - R_b)^2]^{1/2}$ (b = 1,m) is the Euclidean distance

$R_1 = (0,0,\ldots,0)$

$R_b = (p_{1b}, p_{2b}, \ldots, p_{nb})$ (b = 2,m) are defined by equation 3.2

While it has not been proven formally, the allowable unit separation between points is hypothesized to be minimally bounded on the numerical precision of the discretization process employed to transform physical embodiment into machine representation.

E. SUMMARY

A powerful nonlinear transformation has been developed which provides a one-to-one mapping from n space to m space, where typically (m<<n). The effects of this transform on the

information content and cluster formation in the representation are presented in the following chapters.

IV. TRANSFORMATION EFFECTS ON CLUSTERING

The number of valid reference points (R_b) has been shown to be infinite. Implicit in this fact is the choice of transformations is also infinite. Given this infinite selection of transformations the objective is to determine the transform which provides the greatest degree of class clustering and interclass separation. Measures of improvement are discussed as a means to accomplish this objective.

The conversion of pattern space into distance space is presented geometrically to provide some insight into the transformation process.

A. GEOMETRIC INTERPRETATION OF CLUSTER FORMATION IN DISTANCE SPACE

The unique mapping from n space to two space is a function of the (n - 1) space tangent hypersphere generated by the intersection of the radii d_1, d_2. Recall from figure 3.2 that the "direction" in which the tangent surface slices through data space, the range of values for the curvature of the surface of the tangential hypersurface over pattern space and class sample point dispersion relative to that "direction" shape the clustering and class separation which

occurs.

The change in curvature of the surface of the tangential hypersphere is locally minimal but over the entire range of the data space may be quite extreme, depending on the value of reference point two. Points in relatively close proximity will have approximately the same tangential curvature (figure 4.1a). Points which are relatively distant and clustered perpendicular to the tangent line will experience a greater change in curvature between the points (figure 4.1b). Points relatively distant but having nearly the same tangent curve will map in close proximity in two dimensional distance space (figure 4.1c). Points in relative proximity to each other, in the sense their tangent curves are in relative proximity, become more tightly clustered in distance space. The physical reality of this fact is that the points in relative proximity to one another are nearly equal in distance to reference point two.

The concept of relative proximity does not allow total deinterleaving of sample points of different classes. However, any class separation which does exist in any one or more dimensions in pattern space can be enhanced in distance space with the correct choice of reference point two. This enhancement is especially noticable when distance squared instead of distance is utilized to determine an element of the m space vector since distance squared emphasizes the maximum differences between elements of the vectors (X_i -

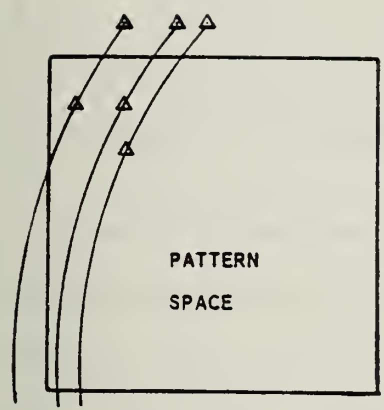
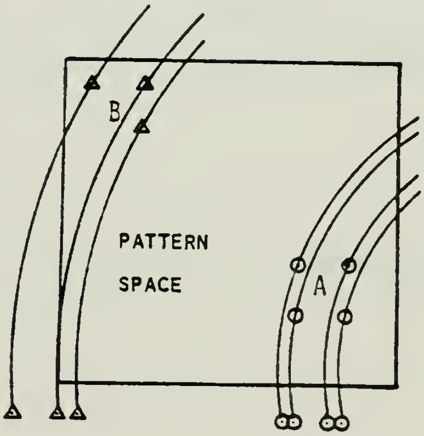


Figure 4.1a Similar tangent arcs passing through a cluster.

Figure 4.1b Variance in curvature

The curvature of the tangent arcs slicing through cluster A is greater than that of the arcs passing through cluster B.



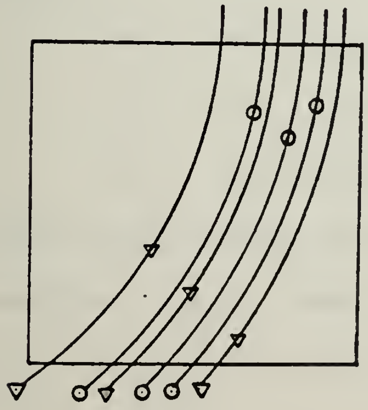


Figure 4.1c Mapping effects on different classes with similar tangent arcs.

R_b).

B. MEASURES OF INFORMATION CONTEXT

One criterion for determining the degree of class clustering and interclass separation is to compare the intraclass variability to the interclass variability. Duda and Hart [1] measure this variability in terms of the scatter within classes (S_W) and the scatter between classes (S_B). Their technique will be utilized in computing a information context measure (I_q), the ratio of (S_W/S_B) in q dimensional space. This ratio is computed in the following manner.

A q dimensional sample mean vector for class i is computed as

$$M_i = \frac{1}{w_i} \sum_{X \in X_i} X \qquad \{4.1\}$$

where

M_i is the mean vector for class i

w_i is the number of samples in class i

X_i is a member of the set of samples in class i

The within class scatter (S_W) is the sum of the within class scatter for each class. The within class scatter is

calculated as the distance squared from a class mean to all points in that class.

$$S_W = \sum_{j=1}^{c} \sum_{i=1}^{w_i} (X_i - M_i)(X_i - M_i)^T \qquad \{4.2\}$$

where

c is the number of classes

i is the class index

$(X_i - M_i)^T$ is the transpose of $(X_i - M_i)$

To calculate the scatter between classes (S_B), the total mean for all classes must be determined.

$$M_T = \frac{1}{z} \sum_{i=1}^{c} w_i M_i \qquad \{4.3\}$$

where

w_i is the number of points in class i

M_i is the mean vector for class i

z is the sum of w_i for all classes

Then the scatter between classes (S_B) is

$$S_B = \sum_{i=1}^{c} w_i (M_i - M_T)(M_i - M_T)^T \qquad \{4.4\}$$

The scatter within decreases as class clustering is

improved. The scatter between increases as interclass separation increases. The ratio (I_q) is computed to measure the interaction of those two facts. Minimizing this ratio when transforming the data samples from pattern space to distance space is a measure of improvement contributed by the transformation. The term miminize is used in a relative sense. The ratio (I_q) will approach zero as the separation between classes increases to infinity or the intraclass scatter decreases to zero. Generally that degree of separation will not be required. By providing a geometrically comprehensiable representation of n space data in one, two, or three dimensions the user may be able to discern a separation which is sufficient without a minimal (I_q) ratio.

The (I_q) measure, when computed as (S_W/S_B) is, in the case of unimodal class distributions, the sum of squared error criterion. For multimodal class distributions and other complex class distributions other optimality measures will be more appropriate. Multimodal class distribution difficulties can sometimes be overcome by redefining multimodal classes into separate classes and applying the (I_q) measure. More complex problems such as dense clusters inside a diffused cluster, for example a sphere within a sphere or interlocking tori, will require different information measuring criteria. These measures are typically application dependent. An example of a three dimensional

cube within a cube is presented as a study of utilitizing the (I_q) measure on complex problems in the following chapter.

C. TRANSFORMATION EFFECTS OF ALTERNATIVE REFERENCE POINTS

The effect of passing the intersection of the two hyperspheres through a pattern space might best be realized in examining a geometrically conceivable example. Consider a three dimensional set of integer points bounded on the interval (0 - 6). This data set is known to contain two classes separated by the plane x = y. The points x = y are members of neither known class. Figure 4.2 illustrates this example. The (I_3) ratio in three space for this data set was computed to be 1.699991. An arbitrarily selected reference point two R_2 = (18,21,-1) yielded a (I_2) = 80.819992. As shown in figure 4.3a this reference point generates a poor mapping solution in terms of clustering since points from distinctly separated classes map adjacent to one another. Note in figure 4.3b that the poor solution was the result of a reference point two which forced the tangential surface direction to have relative proximity across classes. A better choice of R_2 would force the intersection to be nearly parallel to the known class separation. Figure 4.4a shows the results of more carefully selected R_2 = (-999,999,1). Here the (I_2) ratio in two space was .200012.

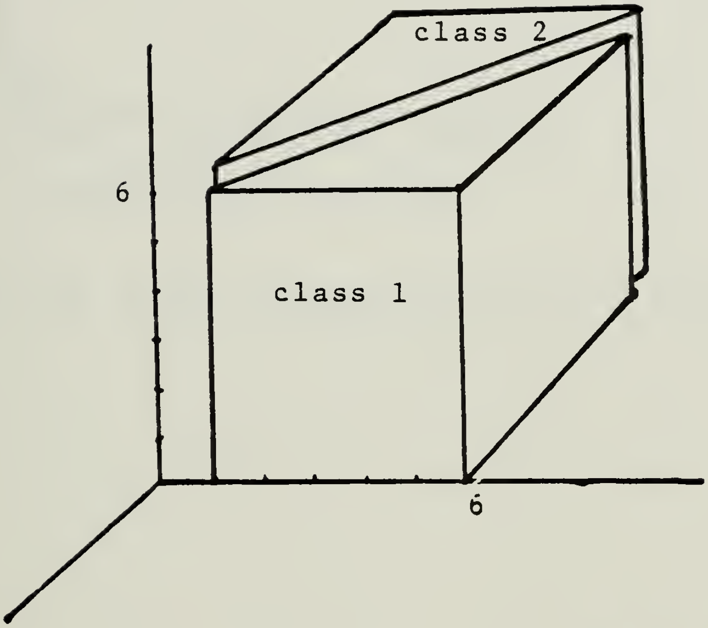
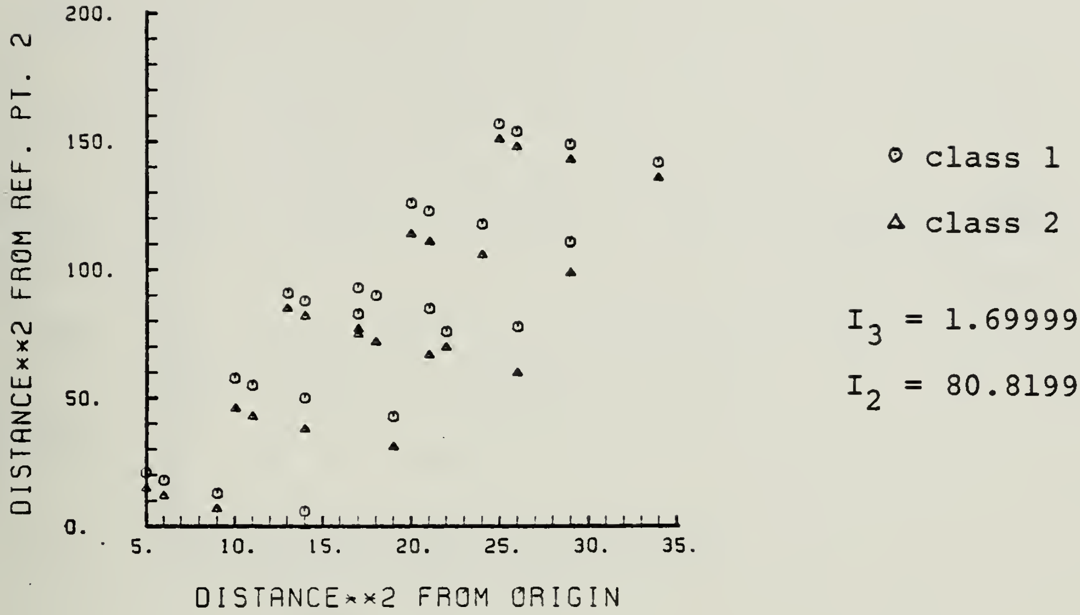


Figure 4.2 A bisected cube

$I_3 = 1.699991$



$I_3 = 1.699991$

$I_2 = 80.819992$

Figure 4.3a A poor mapping solution to the bisected cube

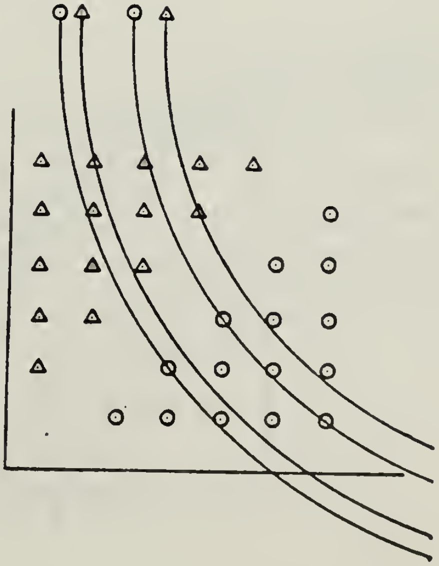
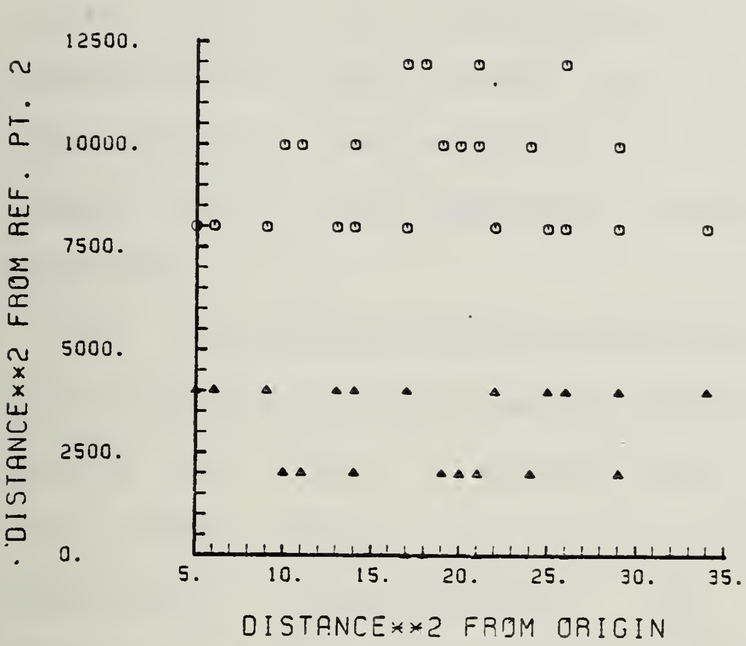


Figure 4.3b A mapping solution which forces the tangent curves across clusters.



$I_3 = 1.699991$

$I_2 = .200012$

Figure 4.4a A nearly optimalmapping solution

Figure 4.4b A mapping solution which separates classes and enhances class clustering.

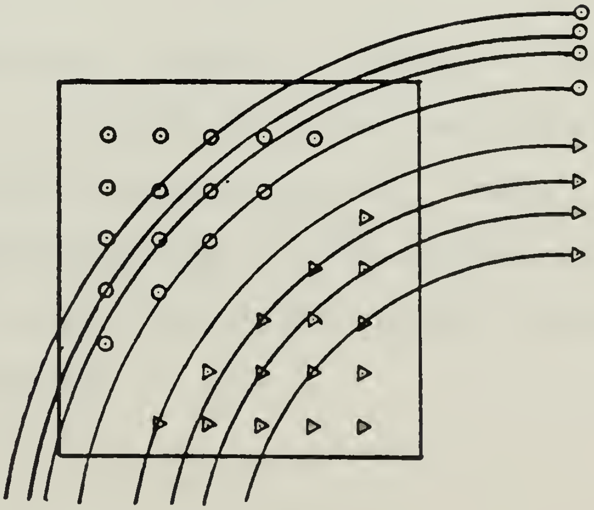


Figure 4.4b is a presentation of this data in two dimensional distance space. This R_2 point enhanced the separation to the extent that the means are spread must further apart. The separation between classes has been enhanced.

An iterative processing technique must be utilized when class separation is not readily apparent. In most cases, varying the value of reference point two toward a minimum (Iq) proved highly successful in finding a good two dimensional representation. The term good implies sufficient. A sufficient ratio must be user defined. Figure 4.4c is a transformation of the example utilizing R_2 = (40,1,6) to yield a (I_2) = .748836. This ratio is not minimal but certainly can be preceived as sufficiently separating the classes.

D. SUMMARY

The location of reference points determine the information context of the distance space representation. When the distribution of classes within pattern space is unknown, an iterative technique based on some optimization criterion can be utilized to locate reference points which provide sufficient transformations.

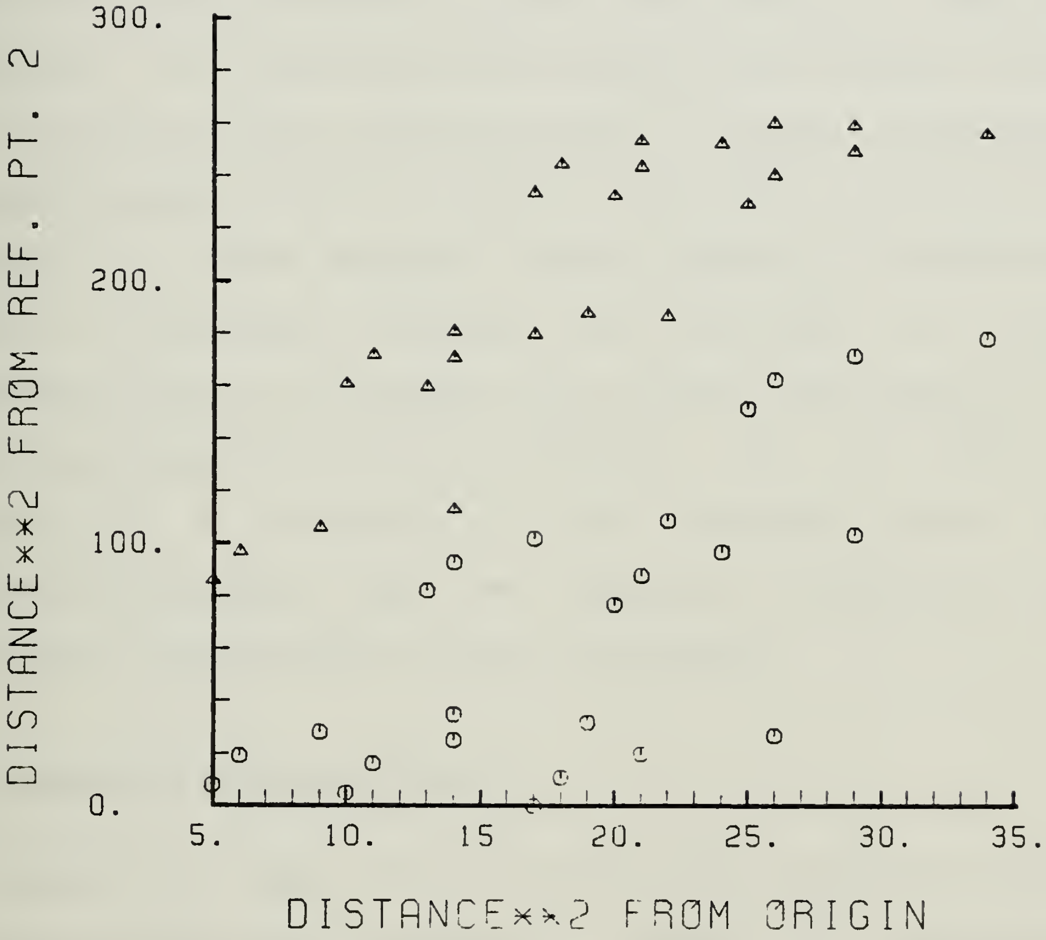


Figure 4.4c A sufficient ratio transformation

$I_3 = 1.699991$ $I_2 = .748836$

V. EVALUATION PROCEDURE

Two different experimental procedures were employed in evaluating the performance of this dimensionality reduction technique. In developing this procedure the initial method was to locate a valid reference point (R) before performing a transformation. This was done to ensure the transformation provided a unique mapping. After deriving an analytical method of partially bounding the locations of valid reference points this method was reversed. The search for a sufficient transformation was conducted and, if required, followed by a validation of the reference point. The validation procedure will be discussed followed by the sufficient transformation search procedures.

A. REFERENCE POINT VALIDATION

Assumption three of section 3C states a reference point (R) exists which will manipulate a circle of intersection such that assumption two of that section will be true for all lattice points in the pattern space defined by assumption one. Without a closed form analyical derivation for validating reference points the only way to validate a reference point was to transform the data into distance space and verify a unique mapping existed. Confirmation

required an exhaustive comparison of each data point with every other data point to prove a unique mapping.

The computational complexity involved in performing reference point validation increased exponentially as the number of dimensions in pattern space increased linearly. This was as expected by the "curse of dimensionality".

The SEARCHR2 computer program contained in this thesis is an implementation of equation 3.7. This algorithm computes the two sets of constant differences , $(D_{1i}^2 - D_{1j}^2)$ and $(x_j - x_i)$, and stores them in array data structures. The alternative method of implementing the algorithm would be to forego the arrays and compute each difference as it is required.

The first method, computing and storing the differences, is computationally more efficient in terms of execution time when more than one candidate reference point is to be verified. This is because the difference matrices need be computed only once. The tradeoff for execution speed is memory storage. The memory requirements increase exponentially as the number of dimensions. The memory requirements can be decreased to some extent by requiring a symmetric pattern space. The differences of $(x_j - x_i)$ is the negative of $(x_i - x_j)$. This implies only (n * (n - 1)) / 2 differences in lieu of (n) differences for each of the two types of differences computed. Table 5.1 vividly demonstrates the "curse of dimensionality" and its effects

on memory storage required for various pattern space configurations.

Table 5.1 Memory requirements of program SEARCHR2 for various pattern space configurations.

DIMENSIONS	RANGE	NUMBER OF LATTICE POINTS	TOTAL MEMORY REQUIRED FOR BOTH ARRAYS (BYTES)
1	0 - 5	6	120
2	0 - 5	36	5670
3	0 - 5	216	301860
4	0 - 5	1296	14265720

In this table, memory requirements for the difference arrays are based on the FORTRAN G programming language as implementated on an IBM 360/67. Single precision, four byte numerical representation is assumed.

The memory allocation requirements can be overcome by computing the (n * (n - 1)) / 2 differences as they are required within the program flow. However the execution time will dramatically increase as the pattern space configuration increases in dimension. Only by optimizing to the greatest extent possible might an acceptable program execution time be generated.

The need for this algorithm has been obviated to some

degree by the development of an analytical derivation of a minumim bound on reference points. This algorithm is still required when validating points inside the minimum bound as noted in section 3C.

An alternative to circumvent both of these problems would be to evaluate a reference point only over the set of known data samples rather than an exhaustive pattern space. This technique would cure the curse of dimensionality but would not guarantee assumption three for all points in pattern space. This concept might be suitable in low noise situations.

This possibility brings to light the consideration that a unique mapping may not necessarily be required. What would be desirable, but much more difficult to define, is a transformation which does not permit samples from different classes to map into the same feature space point.

B. TRANSFORMATION EVALUATION

The study of transformation effects progressed from simple synthetically derived cases in two and three dimensions into complex real data in 10 and 32 dimensions. All cases were mapped into two dimensional distance space. Research was first performed in geometrically conceivable spaces in an attempt to gain the greatest amount of insight. All studies were performed on an IBM 360/67. Transformation

numerical results were produced as well as two dimensional Versatec plots of the resulting distance space representations.

In performing the various case studies the evaluation objectives were to :

1. gain an understanding of transformation effects;
2. confirm that the (I_q) ratio provided a valid measure of transformation performance;
3. determine methods of locating reference points which provide a sufficient transformation;
4. show that real data can be successfully transformed into a lower dimensional representation and still retain information context.

Some insight gained in pursuing objective one has been detailed in chapter IV. Further illustrations will be provided in the following chapter. The chapter on results will document the usefulness of the (I_q) ratio and acknowledge some of its weaknesses.

The methods of locating sufficient reference points were constantly being refined in processing the various case studies. Initially it was thought that the optimium reference points would exist only along an axis. This is most definitely not true. The search procedure evolved into an iterative processing method. This method rated various reference points solely on their resulting (I_q) value. The procedure was to minimize the (I_q) ratio since the smaller

in value (I_q), the more information context present in the feature space representation. The maximum value of the magnitude of the reference vector is unbounded. To limit the search region the user arbitrarily selected a maximum bound magnitude. Utilizing either the positive or negative bound, a single component of the reference vector was varied by some interval to the opposite bound while holding all other components of the reference vector constant. This allowed the user to observe effects of various values of this vector component. The effects were judged by comparing the various values of (I_q) generated. The minimum (I_q) was selected and the procedure started again with a smaller interval centered around the component value which generated the minimal (I_q). This procedure was continued until a minimal (I_q) ratio had been reached with that component. The process was repeated for each component in the reference vector.

For simple cases of three or four dimensions combinations of two or three minimal component values were tested with all other values held constant. This test occasionaly provided useful results but was not consistent. After each component of the reference vector had been tested individually, all minimal (I_q) producing components were combined together and tested. For all test cases, this procedure always yielded the lowest (I_q) measure. When all (I_q) measures were compared against their plots the minimum (I_q) always had the "best" appearing plot in terms of class

clustering and class separation.

As of this report this iterative processing procedure is the methodology recommended. However, this procedure is not without faults. The test and evaluation procedure requires testing each component of the n dimensional reference vector. This is a minor bother in low dimension pattern spaces but becomes quite awkward in high dimensional spaces. The 32 space reference vector would have required 34 computer runs to claim a satisfactory search for a sufficient reference point. Secondly, it was discovered, not unexpectedly, that local minima of (I_q) exist within the bounded set of possible reference points. At present there is no closed form solution for finding the optimal reference point. The only way to overcome problems of local minima is extensive testing. Thirdly, if a sufficient reference point is found to exist below the minimum boundary for reference point two, the SEARCHR2 program will be required to validate the reference point. This complicates the situation even further with all the limitations of that program.

C. Implementation tradeoffs

This technique provides the researcher or engineer a choice of alternatives. One alterative will require a certain amount of not inconsequential time and effort to find a reference point or points to provide a sufficient

distance space representation. But the fruit of that effort will be the ability to perform the remaining portions of testing and training in distance space. Significant here is that the time consuming portion of the effort is devoted to the training phase. This is typically not the phase which is time critical. The distance space representations are less computationally complex. Hence, the testing phase will benefit from the reduced complexity with decreased execution times. This will be especially useful in real time decision making applications where the testing phase is time critical.

The second alternative is to forego the time consuming effort in training but with a commensurate increase in computational complexity in the testing phase.

The researcher or engineer must judge which is the most cost effective for his application. The following chapter will hopefully provide some insight into what is required if the first alternative is selected.

VI. EVALUATION RESULTS

In evaluating the transformation seven test cases were studied. The evaluation proceeded from simple three space problems to a complex 32 dimensional real data problem. Discussion of each case comments on how the data samples were derived, the procedures in evaluating the case studied, and the results. Graphic illustrations are provided as appropriate. The appendices contain complete documentation on the progression of testing for all cases studied.

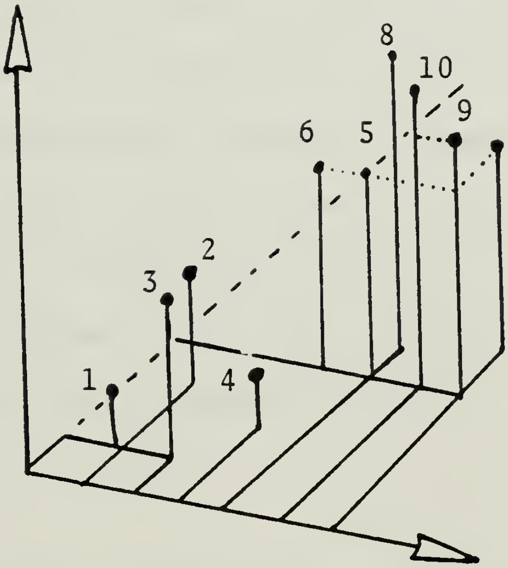
A. CASE 1 : A THREE CLASS THREE SPACE PROBLEM

A simple problem was first attempted to gain insight into the transformation process. Figure 6.1a illustrates the heuristically derived points in pattern space. Appendix A contains a listing of the points which make up each class. The transformation was from three space to two space. In three space (I_3) = .2319393. The best (I_2) ratio achieved, in a less than exhaustive search, was 0.07986128.

Various linear combinations of reference point twos (R_2) were tested to observe the transformation effects on pattern space. The distance space representations were plotted with the X axis as distance to the origin squared and the Y axis as distance to R_2 squared. It is interesting to note in

Figure 6.1a A three class three space problem

$I_3 = 0.231939$



class 1 (1,2,3,4)

class 2 (5,6)

class 3 (7,8,9,10)

figures 6.2b - f the results developed utilizing various linear combinations of an R_2 point. Observe that the relative proximity of the classes is unchanged while the relative positions within the classes are a function of the reference point two utilized in the transformation. This is to be generally expected as the within classes relationships will change as the reference point two changes. This is also a function of the fact that the distance to the origin never changes while the (d_2) values vary as the (R_2) vary.

This last observation suggest an examination of moving reference point one from the origin in an attempt to improve class separation. While it may be a valid concept the transformation becomes much more complex in the process. This examination is suggested as a topic for further research.

B. CASE 2 : A THREE DIMENSIONAL BISECTED CUBE

A three dimensional cube of integer lattice points was developed for this study. This case was previously discussed as the geometrically conceivable example in section IV C. Figure 4.2 illustrated the three space configuration of the classes.

This case has some interesting complications. In pattern space the (I_3) ratio is 1.699991. The class means are :

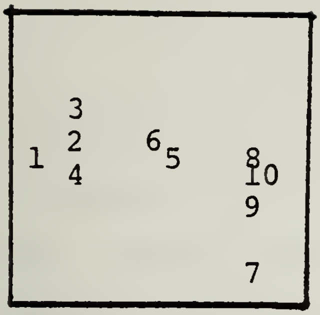


Figure 6.1b $I_2 = .096678$
$R_2 = (-1,1,6)$

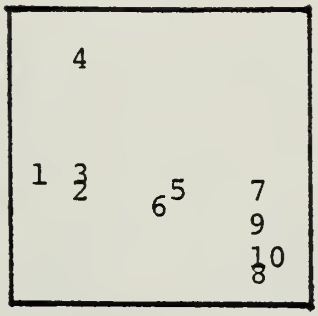


Figure 6.1c
$I_2 = .094691$
$R_2 = (6,1,-1)$

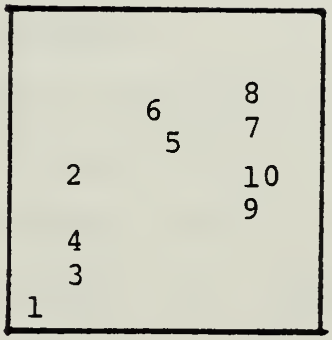


Figure 6.1d $I_2 = .212315$
$R_2 = (-1,21,8)$

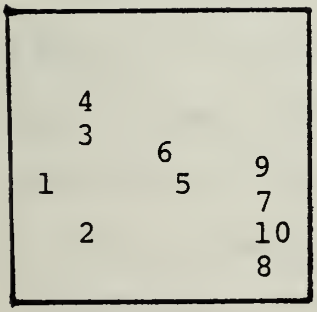


Figure 6.1e $I_2 = .080043$
$R_2 = (6,-1,1)$

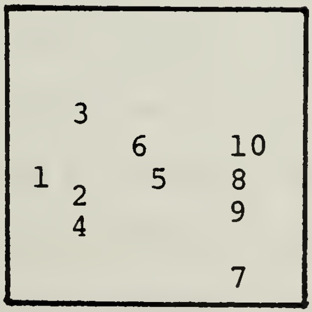


Figure 6.1f
$I_2 = .079861$
$R_2 = (1,-1,6)$

Variations in within class relationships as R_2 changes.

$I_3 = 0.231939$

	(x , y , z)
class 1	(1.66, 3.33, 1.5)
class 2	(3.33, 1.66, 1.5).

The scatter within classes (S_W) are equal. An (I_q) ratio greater than one implies the distance between the two class means is less than the average distance between a class mean and the samples within that class. Even with the (I_q) ratio greater than one there exists a distinct linear separating boundary between the classes.

Appendix B contains a listing of the data points in each class and a summary of the iterative processing steps for this case. Ten iterations were required to achieve an "optimal" value. As noted in chapter IV a sufficient solution was obtained on the fourth iteration. This demonstrates the value of having a two dimensional picture of the data on which to make judgements about that data.

It occured to this researcher, upon examining the results of iterations six and seven (see appendix B), that the minumum value was symmetric in magnitude in the X and Y components. Iterations 7 - 12 attempted to exploit this fact by testing beyond the user defined bound of 999 in the X and Y components of the vector. This proved quite successful as figure 4.3a illustrated. By symmetrically increasing the values of the X and Y components the (I_2) ratio was minimized. As reference point two is moved further from pattern space the change in curvature of the tangent

intersection becomes almost nil. This occurs because the circle of intersection begins to approximate the surface of the sphere defined by the (d_2) distance as shown in figure 6.2. In effect, this forces similar points to map almost linearly into distance space. This fact accounts for the apparent subclusters that exist in each of the classes.

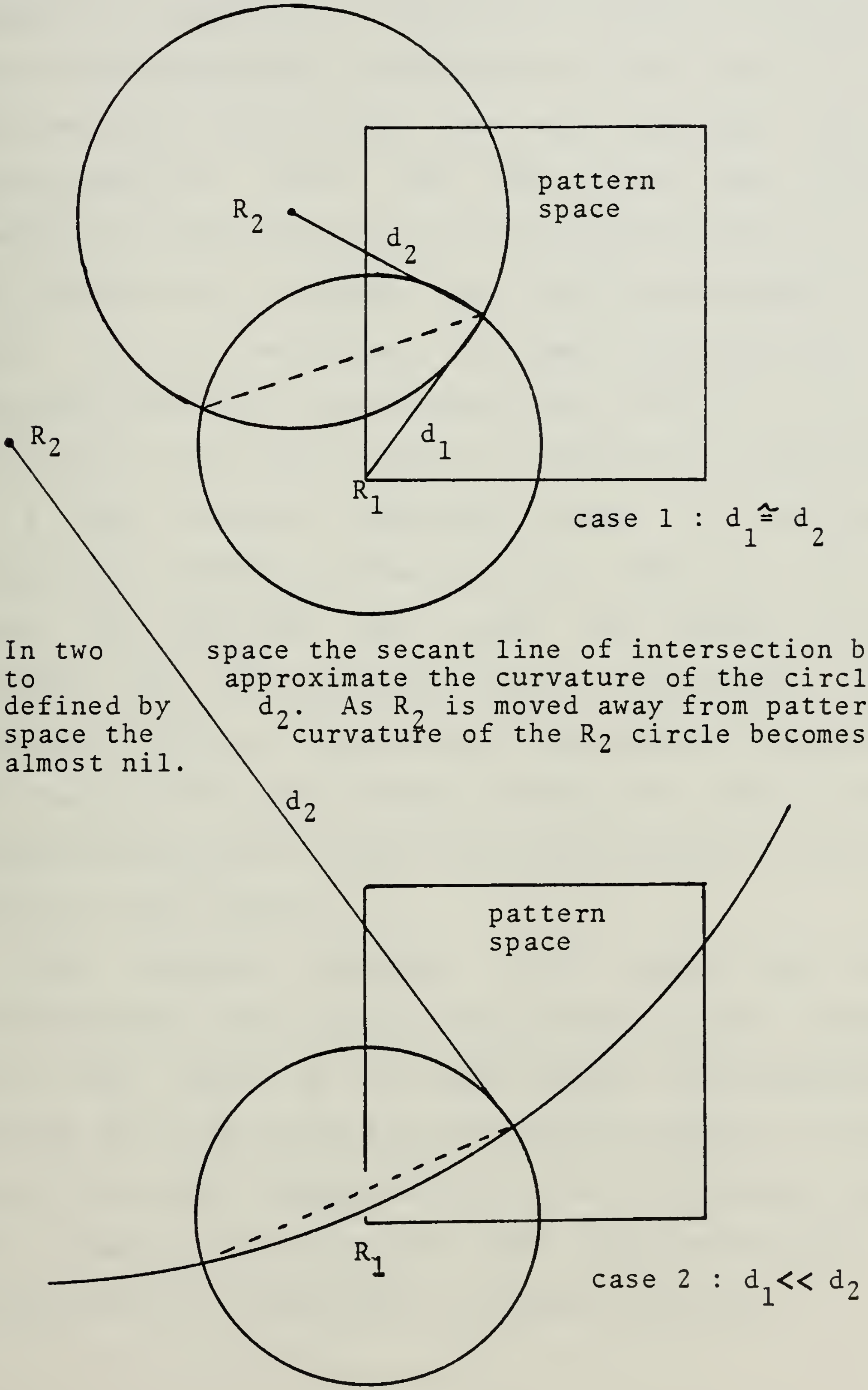
In step seven, the Z components were varied to study their effect on the (I_2) ratio. As can be seen from the data increasing Z in magnitude above one adversely affected the (I_2) ratio. In iterations 9,10,and 11 note that symmetric points in the X,Y plane yield similar, if not equal, results. This can be explained by realizing that the "direction" of the circle of intersection of the various symmetric points and reference point one are very nearly equal. They differ in that the R_2 sphere is located on the opposite side of the R_1 sphere.

This case study proved the most enlighting as to the geometric effects of passing the intersection of the spheres through pattern space.

C. CASE 3 : A CUBE WITHIN A CUBE

This case was selected to study the effectiveness of using the (I_q) measure in a complex problem. Two cubes were generated in a three dimensional pattern space containing integer lattice points in the range (0,0,0) to (5,5,5),

Figure 6.2 Change in curvature of the tangent intersection



In two space the secant line of intersection begins to approximate the curvature of the circle defined by d_2. As R_2 is moved away from pattern space the curvature of the R_2 circle becomes almost nil.

Appendix C contains a complete listing of the data. The outer cube exhaustively surrounded the inner cube. The inner cube mean was (3.0,3.0,2.5). The outer cube mean value was (2.84,2.84,2.5). Thus the outer cube mean value was contained inside the inner cube. The three space (I_3) ratio was 2950.249. The "optimal" (I_2) ratio was 66.218964. Figure 6.3a is the resulting two space representation. The reference point two which yielded this result was (6,1,-1). A listing of all points tested is contained in appendix C.

A more visually appealing result is illustrated in figure 6.3b. Generated from R_2 equals (360,-60,1), the (I_2) ratio was 2574.547. The points are bunched into six clusters. Each group of points can be characterizd by its location on the Z axis in the three space representation. For example, the group located nearest the X axis in the two space representation is all the points located on the Z = 0 plane in three space.

This example provides some insight on how the transformation skews pattern space into distance space. In this three space to two space example, the perspective is changed much like an artist would skew an image to provide a three dimensional perspective on a two dimensional canvas. This suggest there may be some applications for this transformation in the field of computer graphics.

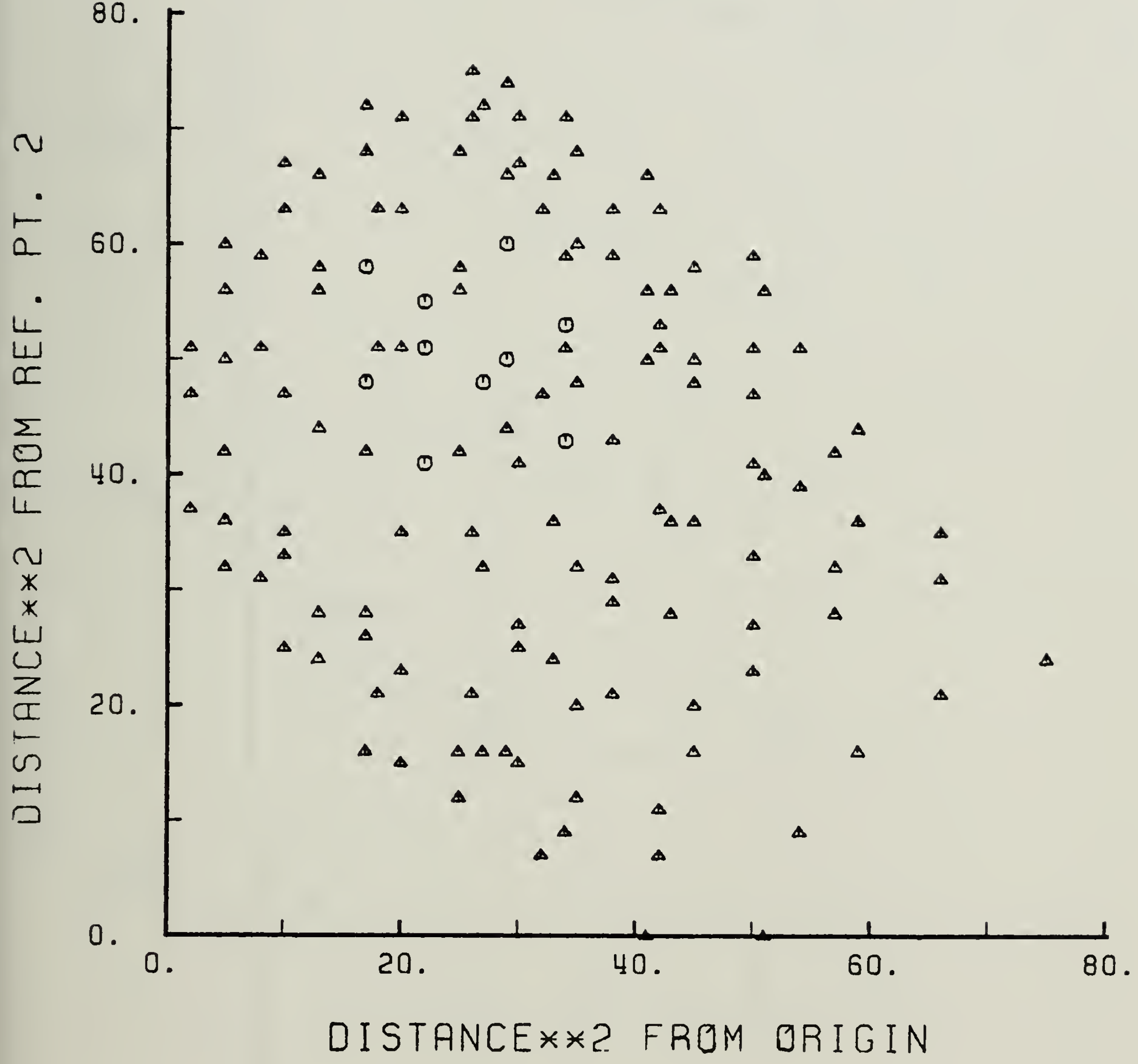
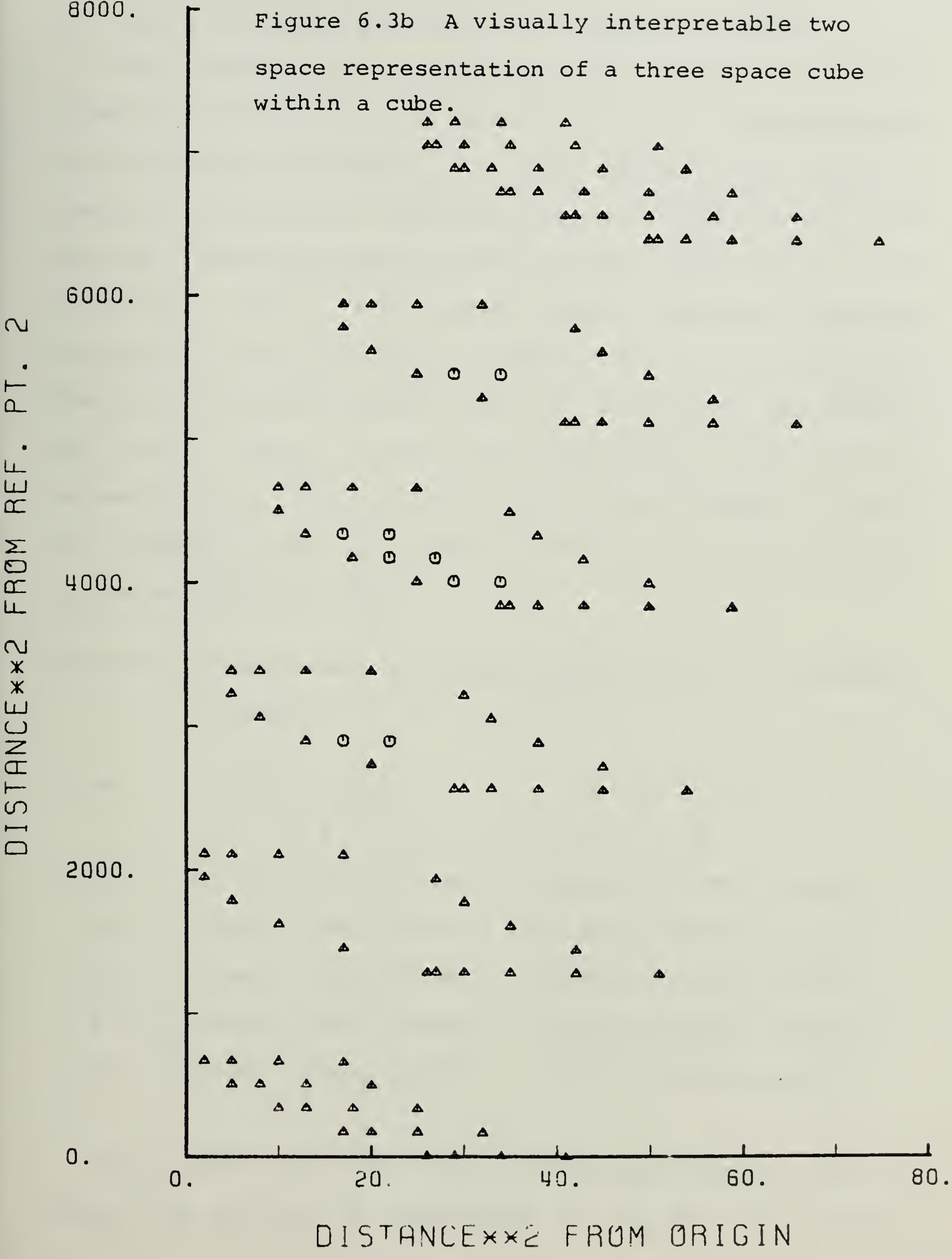


Figure 6.3a A two dimensional distance space representation of a three dimensional cube within a cube.

I_3 = 2950.249 I_2 = 66.218964

Figure 6.3b A visually interpretable two space representation of a three space cube within a cube.



D. CASE 4 : THREE SPACE SERIES OF CONVERGING CLASSES

This study was a series of five cases in three dimensional space in a lattice containing all integer points from (0,0,0) to (24,24,24). The cases differ by the relative location of the two classes in space. The data points for the two classes were generated on a Texas Instruments TI-59 calculator using the random number generator program (ML-15). The class one seed was 2135. The class two seed was 7540. The standard deviation used was 2.0. These parameters were kept constant across the experiment so that the only parameters of variability were the distance between classes and reference point two. Table 6.2 contains the mean samples generated for each class.

Table 6.2 Sample means for three space series of converging classes

case	class 1 mean			class 2 mean		
	x	y	z	x	y	z
1	4.25	3.41	3.08	20.16	19.67	19.58
2	6.08	5.41	5.08	18.33	18.50	17.66
3	8.00	7.50	7.08	16.33	16.50	15.67
4	9.16	9.41	9.08	14.33	14.50	13.66
5	12.08	11.41	11.08	12.33	12.50	11.67

This study was two fold in purpose. One goal was to observe the effects on selection of R_2 as the classes

approached one another. Secondly, an attempt was made to distinguish where the transformation would not longer separate classes.

As before all cases were processed using the systematic iteration procedure delineated in the previous chapter. The following results were obtained.

Table 6.3 I_q results for three space converging classes

case	I_3	I_2	reference point	iterations	figure
1	0.061068	0.014587	(66, 66, 66)	5	6.4a
2	0.106720	0.027066	(63, 63, 63)	4	6.4b
3	0.238156	0.47232	(-99,-540,-99)	5	6.4c
4	0.813121	0.153407	(-99,-540,-99)	4	6.4d
5	32.466873	4.560885	(-150,-900,-150)	3	6.4e

Cases one and two share similar reference points as do cases three and four. Unfortunately, there is not enough samples here to rationalize why this is so. Since the classes vary between cases only by their separation then it will said that the relative proximity between classes does affect the choice of R_2. In case five, the classes shared the same mean. They were highly interleaved as shown by the I_3 ratio of 32.466873. The change in the I_2 ratio is a measure of how much deinterleaving took place in the

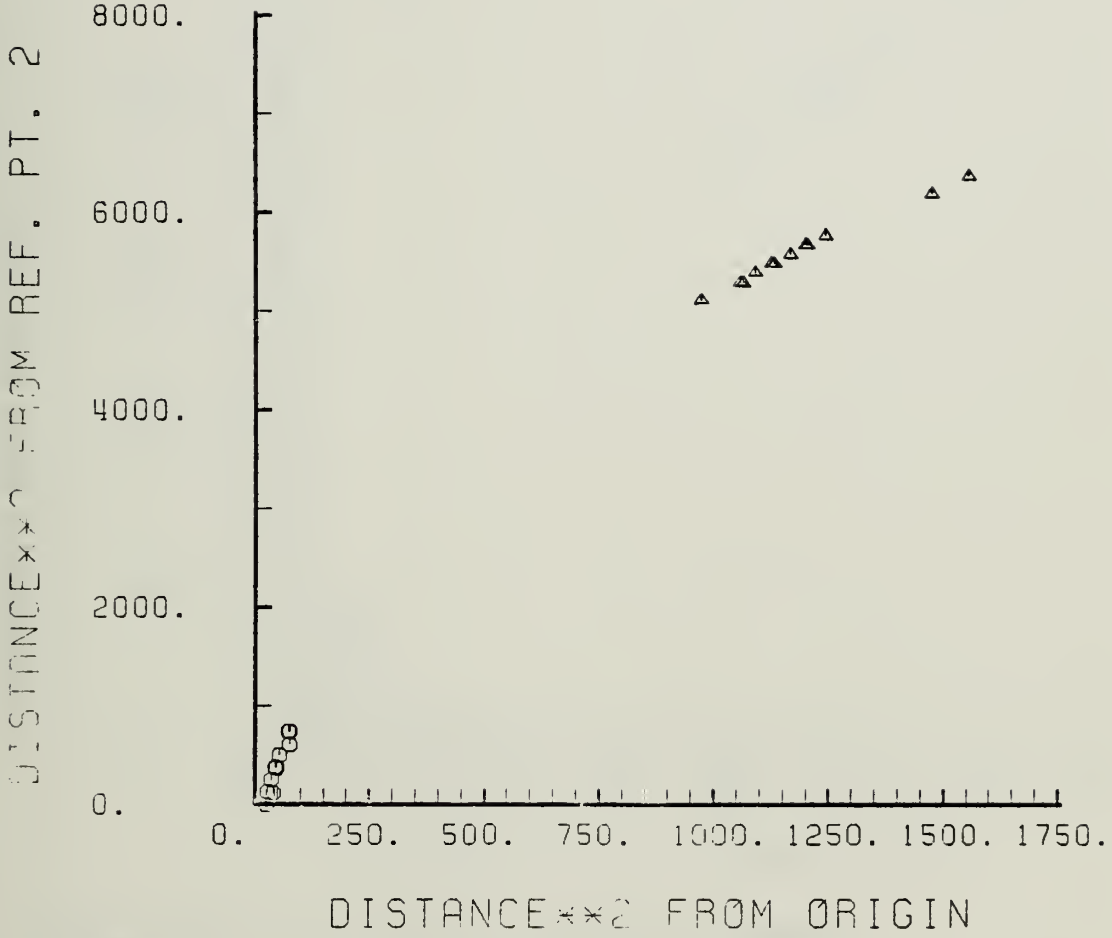


Figure 6.4a Two space solution to case 1

I_3 = .061068 I_2 = .014587

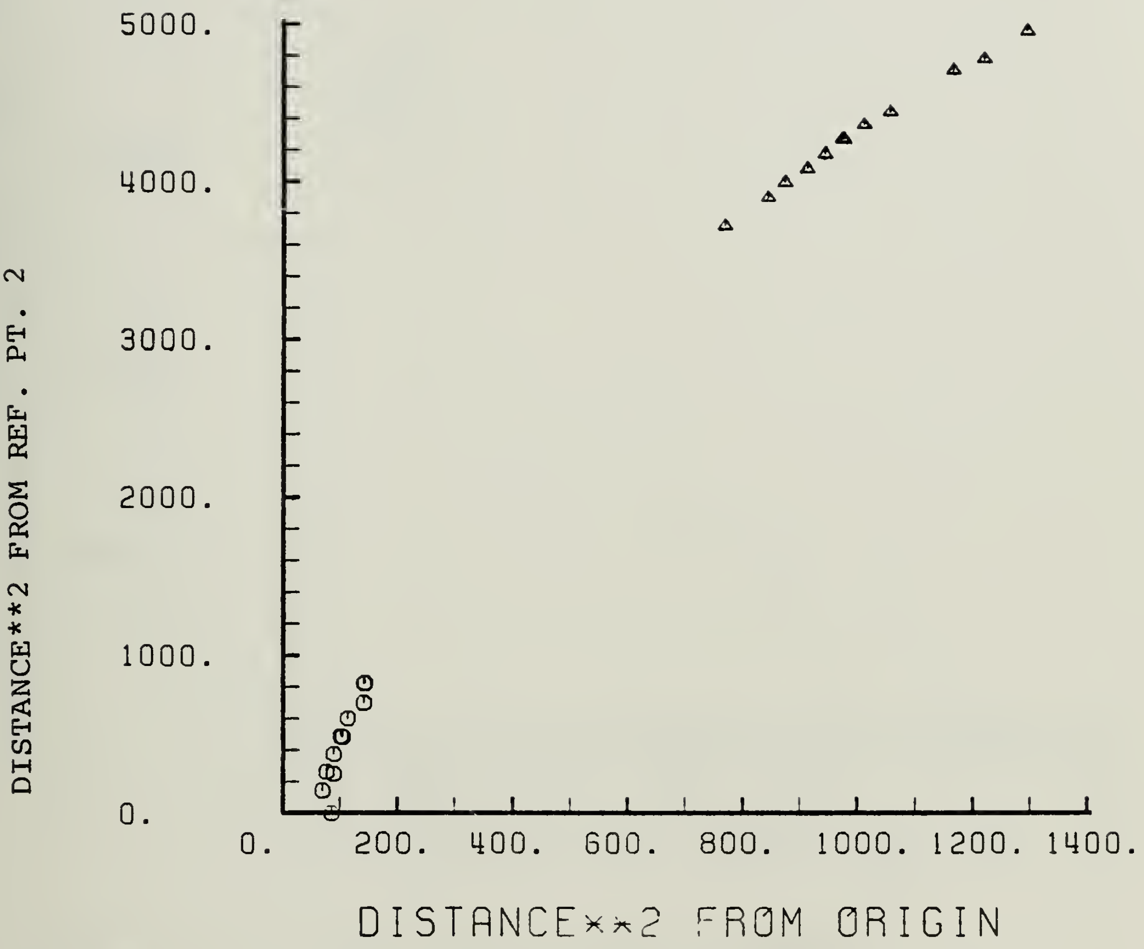


Figure 6.4b Two space solution to case 2

I_3 = .106720 I_2 = .027066

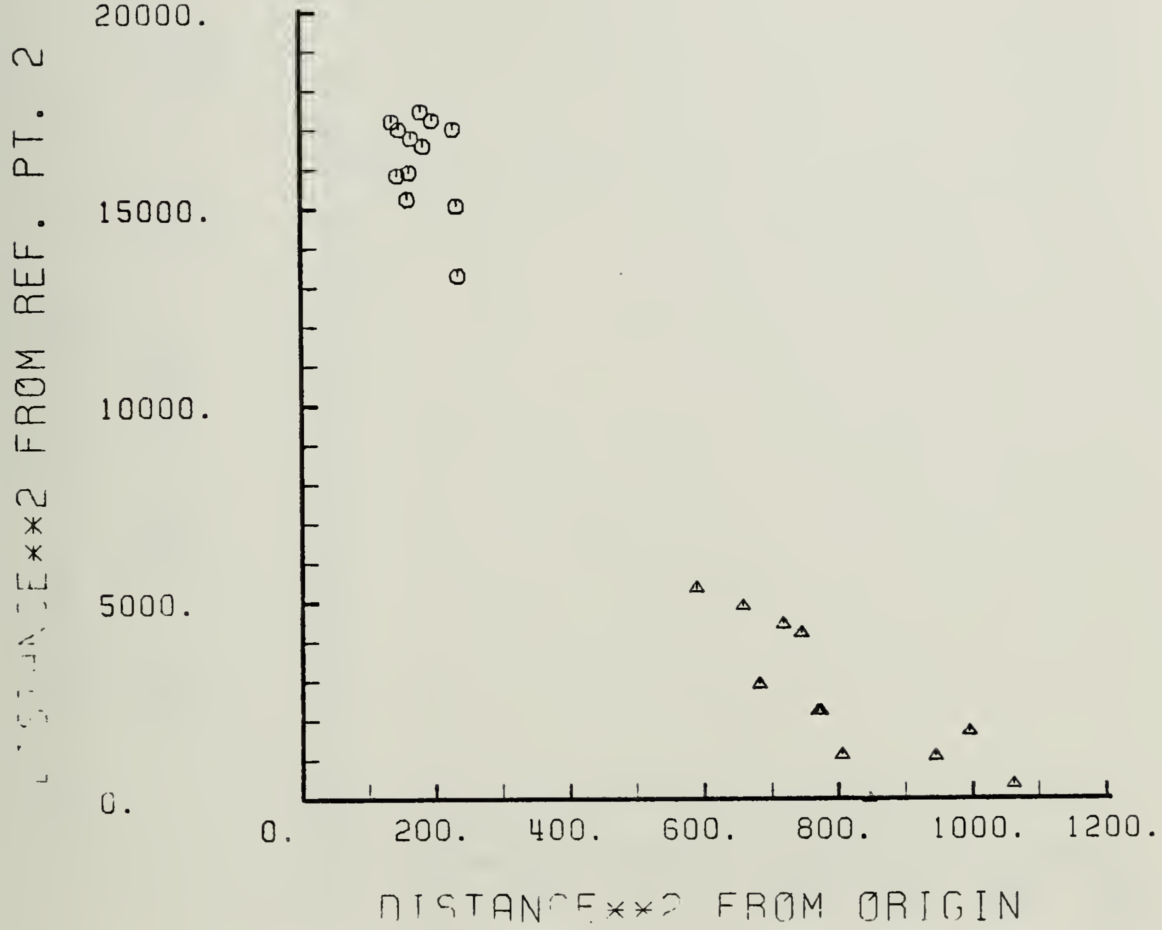


Figure 6.4c Two space solution to case 3

I_3 = .238156 I_2 = .472320

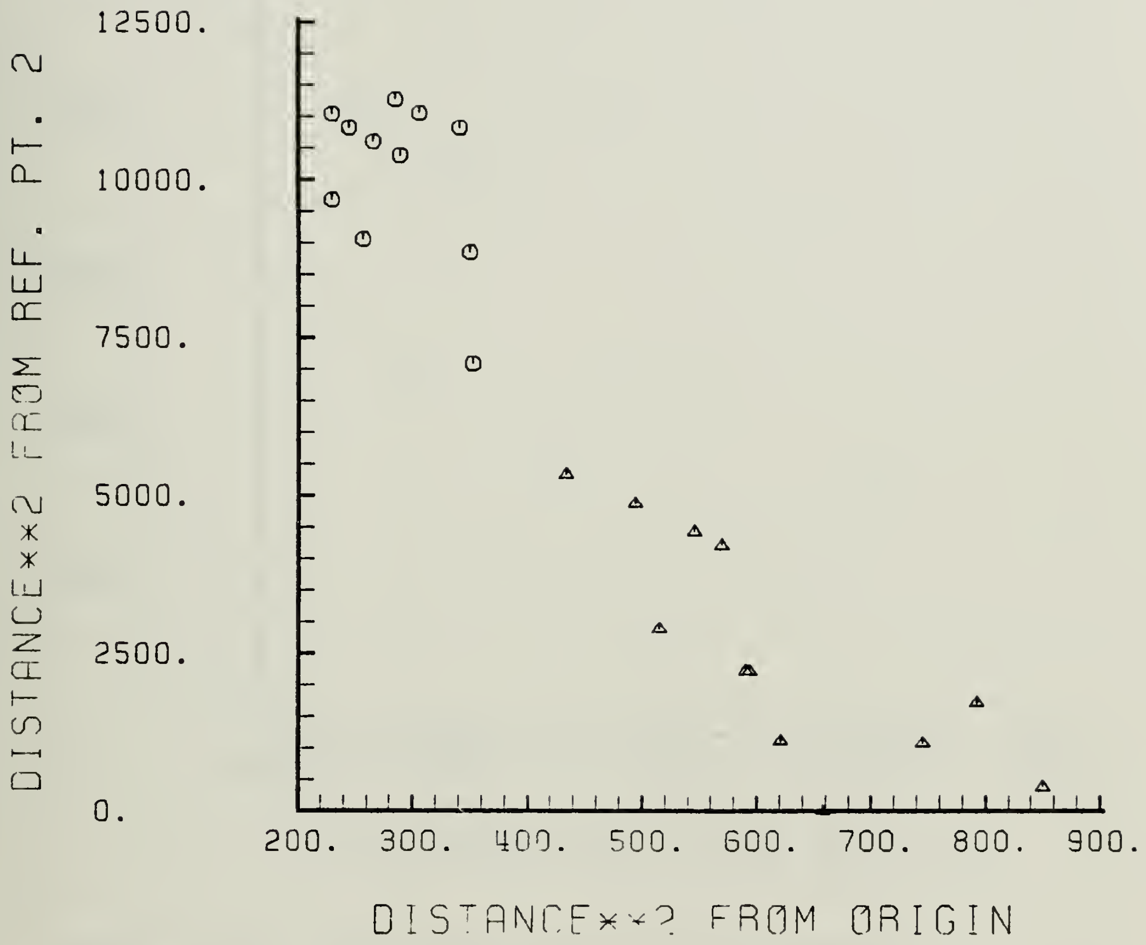


Figure 6.4d Two space solution to case 4

I_3 = .813121 I_2 = .153407

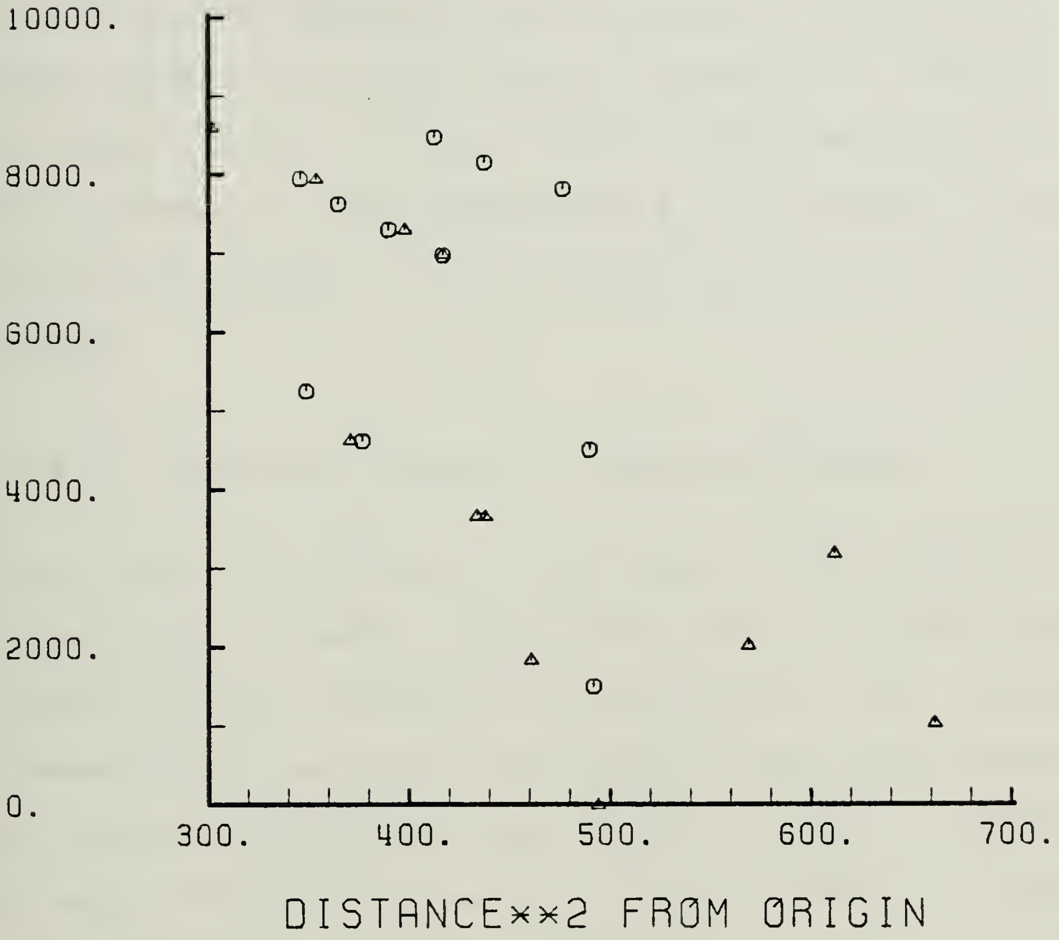


Figure 6.4e Two space solution to case 5

$I_3 = 32.466873$ $I_2 = 4.560885$

transformation. Case five further supports the premise that the relative proximity of the three space data was maintained in that total deinterleaving did not occur. This technique claims to only enhance separation present in pattern space in the distance space representations. To further comment on the behavior of R_2 will require a more exhaustive examination of its performance under similar situations.

E. CASE 5 : TEN SPACE SERIES OF CONVERGING CLASSES

This study is a series of four cases. Pattern space is a ten dimensional integer lattice containing all points from the origin to (19,19,19,19,19,19,19,19,19,19). Two classes were generated in each case. The samples were synthetically derived using the IBM 360/67 and utility program LLRANDOM entry point NORMAL. The seeds were held constant for each class over all four cases. The variance selected was 2.0. The following means were generated for each of the classes. Read a class mean vector as a column in the table.

Table 6.4 Sample means for ten space series of converging classes

case	1		2		3		4	
class	1	2	1	2	1	2	1	2
	4.12	15.43	5.12	14.56	8.62	12.06	9.81	10.06

3.18	15.68	6.00	15.12	8.43	12.62	9.75	10.06
4.06	15.43	5.50	13.31	8.06	12.43	10.31	10.06
4.56	16.50	6.18	14.06	8.06	12.43	10.31	10.06
2.93	15.93	6.73	14.56	8.06	11.93	9.93	9.78
4.50	15.37	5.75	14.06	8.12	12.31	10.31	9.87
4.25	16.18	5.81	14.25	8.62	12.18	10.37	9.12
3.81	16.37	5.62	13.50	8.06	12.00	9.93	9.06
3.75	17.00	6.18	13.81	7.56	11.62	9.62	9.18
4.00	16.12	6.43	14.37	8.00	12.37	10.18	9.75

The goals of this study are similar to the previous study in three dimensions; to observe the behavior of R_2 as the classes approached one another and to attempt to distinguish when the transformation would no longer separate classes. The results obtained are detailed in table 6.5.

Table 6.5 Iq results for ten space converging classes

case	I_{10}	I_2	iteration	figure
1	0.069465	0.004561	1	6.5a
2	0.158126	0.023578	1	6.5b
3	0.594771	0.062027	1	6.5c
4	36.600388	8.294548	8	6.5d

The I_q results were obtained with the following reference points:

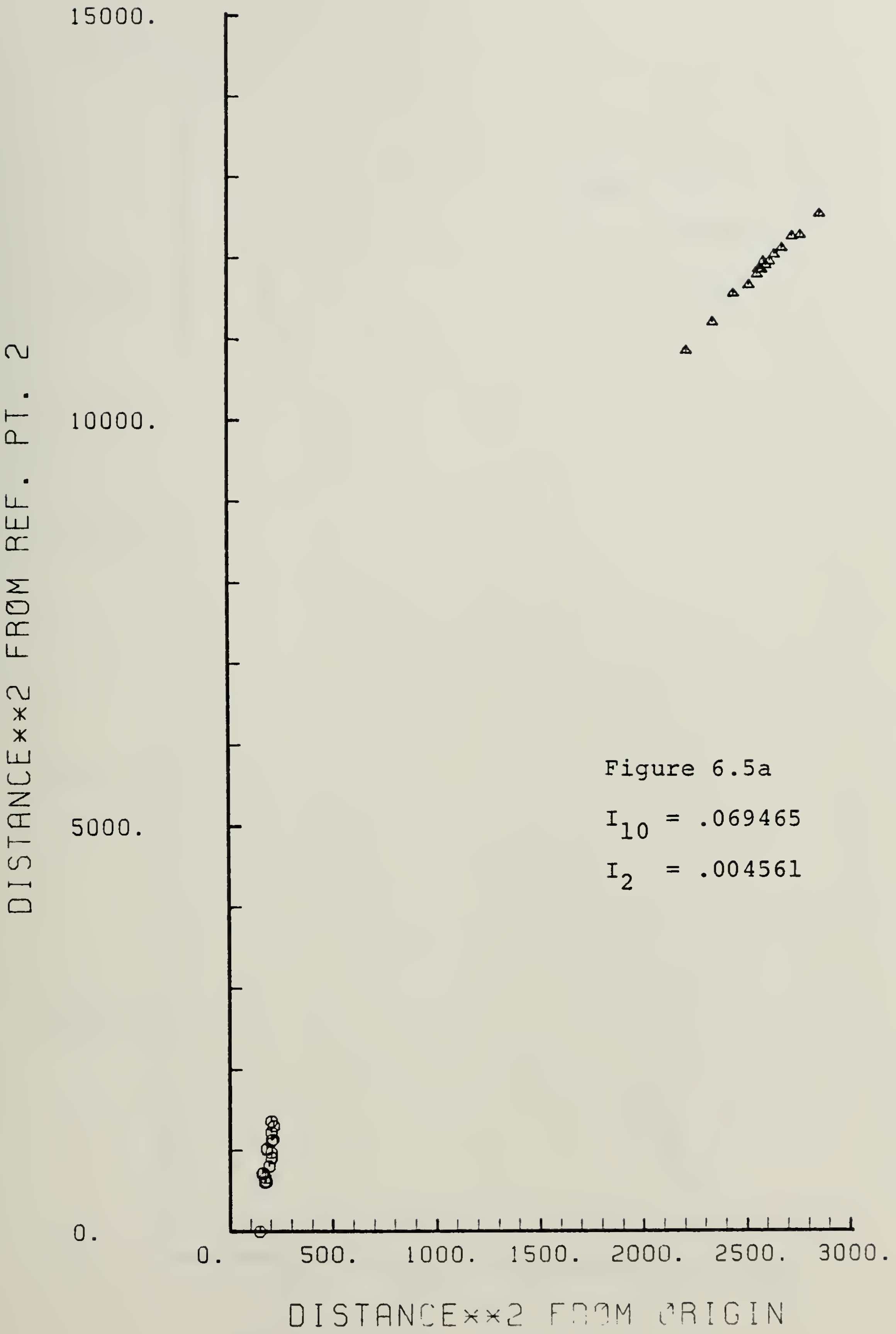


Figure 6.5a

I_{10} = .069465

I_2 = .004561

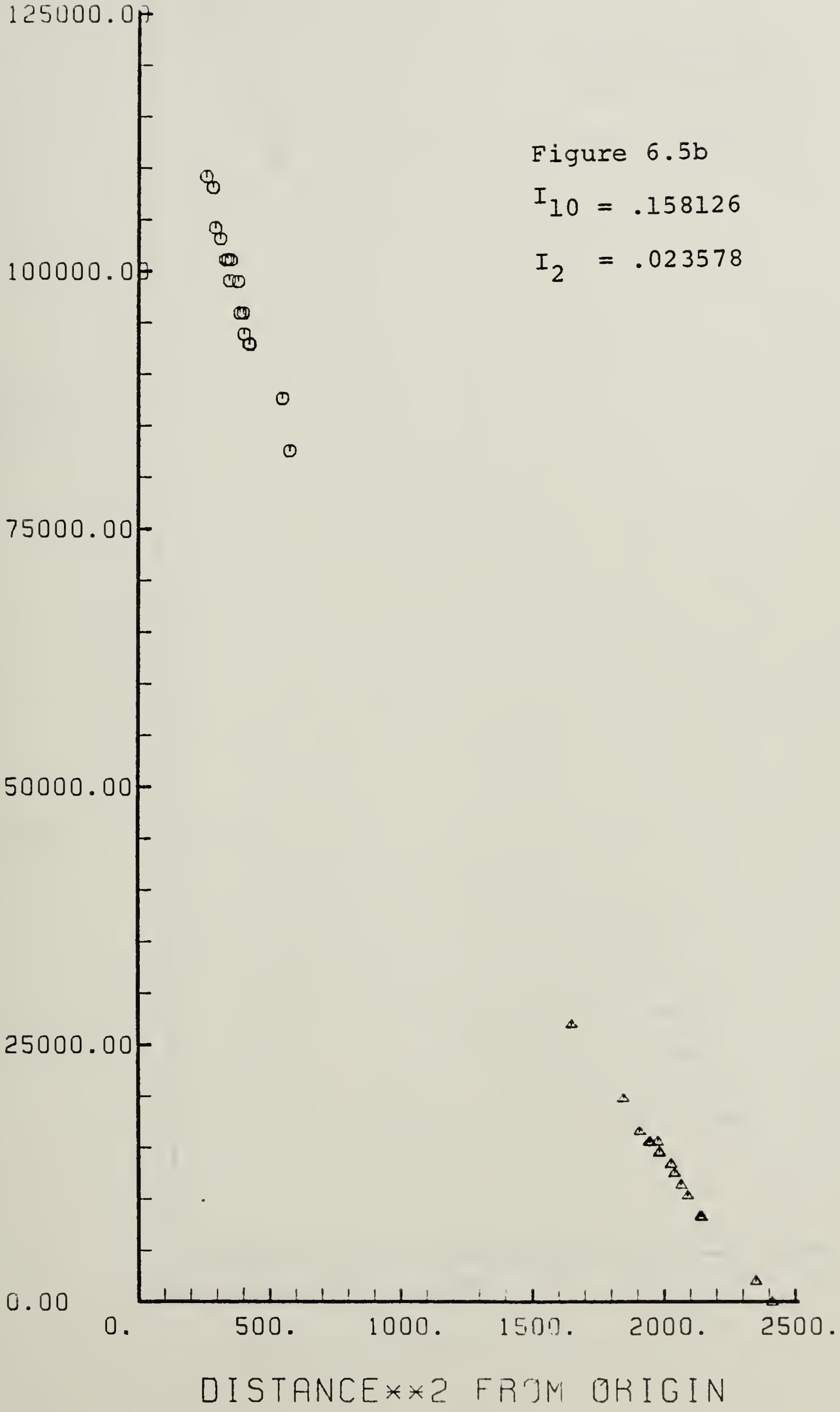


Figure 6.5b

I_{10} = .158126

I_2 = .023578

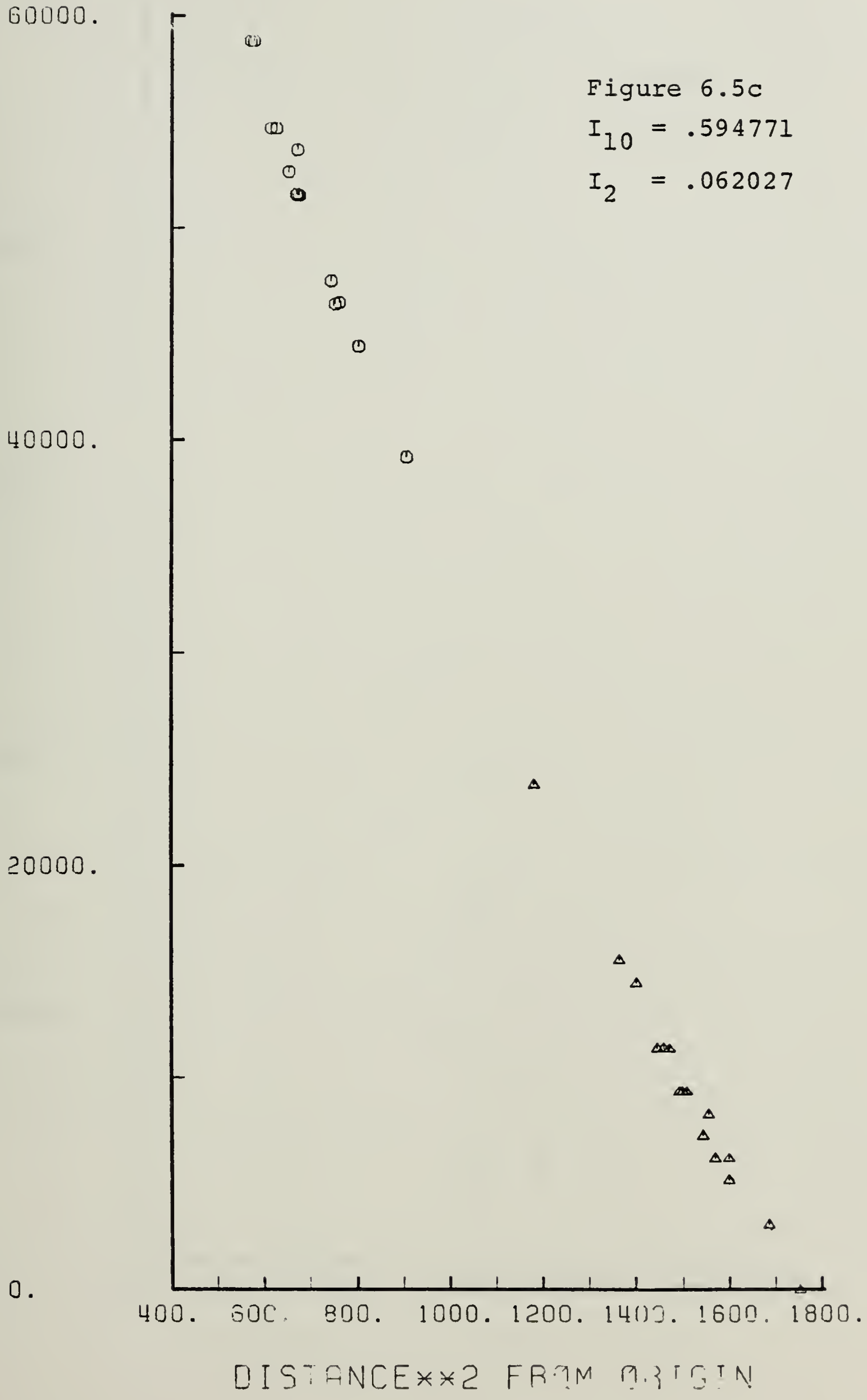


Figure 6.5c

$I_{10} = .594771$

$I_2 = .062027$

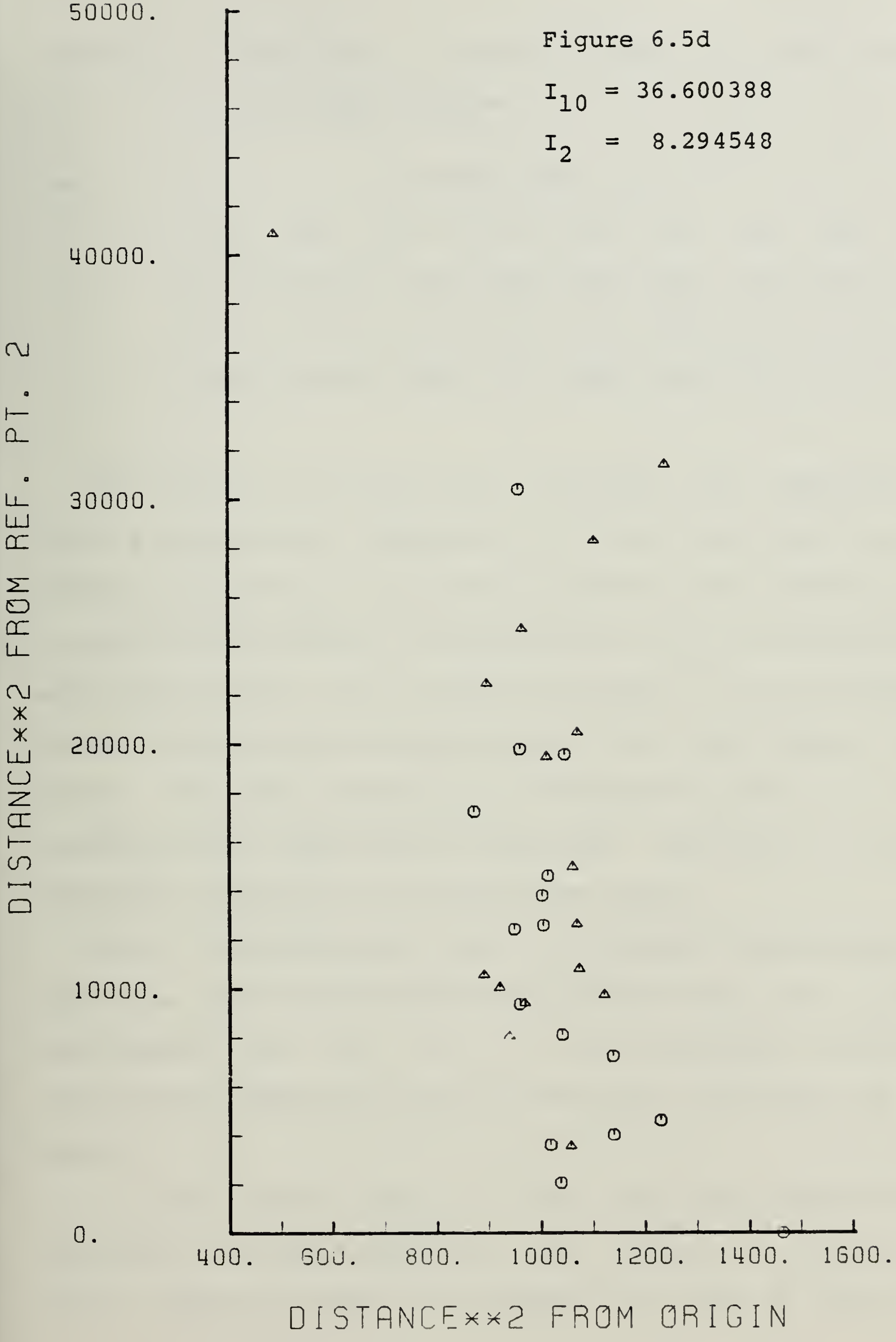


Figure 6.5d

I_{10} = 36.600388

I_2 = 8.294548

Table 6.6 Sufficient reference points for ten space converging classes

case	reference point two
1	(51, 52, 53, 54, 55, 56, 57, 58, 59, 60)
2	(-501,-502,-503,-504,-505,-506,-507,-508,-509,-510)
3	(-501,-502,-503,-504,-505,-506,-507,-508,-509,-510)
4	(999, 800,-821,-219,-111, 900,-999,-411,-511,-611)

In observing that only one iteration was required to find a sufficient separation, it can be said that the inherent separation in the data allowed a wide choice of R_2 to provide a sufficient transformation. It is interesting to note that cases two and three utilized the same reference point. This is the same occurance as with the three space study. In both studies the reference points for the noninterleaved cases were not near the sufficient solution reference points for the interleaved cases.

The transformation had no problem maintaining or enhancing the inherent separation in the data. Only in the overlapping case was an $I_2 < 1$ not achieved. However, a significant reduction in the I_q ratio was achieved in all cases.

In the fourth case of this study the adminstrative complexity of high dimensional data became a burden. To allow sampling over all elements of the R_2 vector ten

iterations should be required. In the evaluation only eight were performed as modifying the individual components of the reference vector became unwieldy. The burden could be essentially overcome with a more automated test procedure at the cost of more computer execution time.

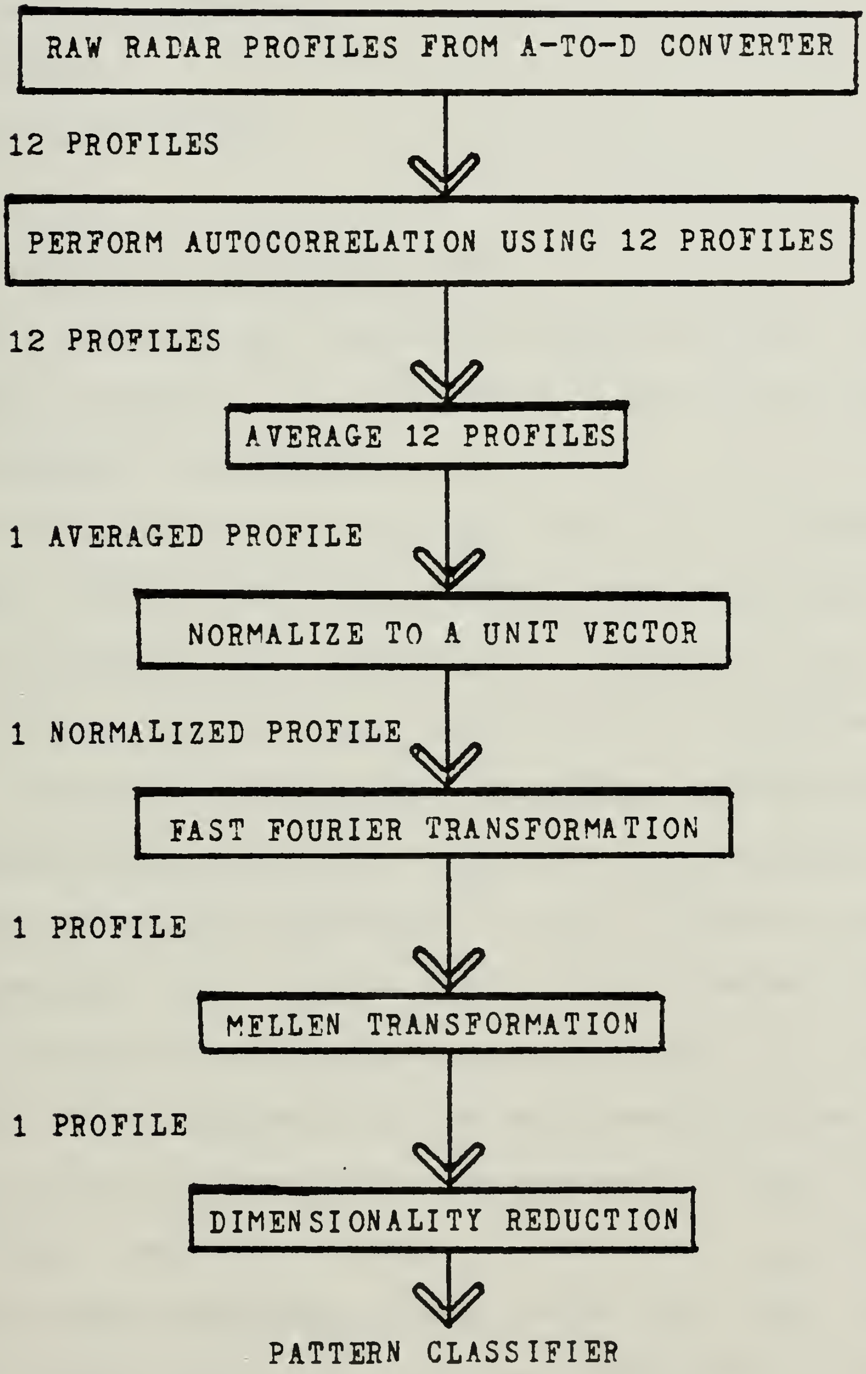
G. CASE 6 : TEN SPACE THREE SHIP RADAR TARGET RECOGNITION DATA

This study was conducted to observe the performance of the transformation on real data in high dimensions.

The data was collected by the Naval Weapons Center, China Lake, California as part of a research effort on radar ship classification techniques. The samples consisted of radar profiles of selected ships. The samples experienced some preprocessing prior to being input to the dimensionality reduction technique. The received radar echoes were taken from the radar receiver and fed through an analog-to-digital converter and eventually stored on magnetic tape. The remaining preprocessing is shown in figure 6.6.

The samples were gathered in 512 dimensional space. For the purposes of this study only the first ten components of each sample were utilized. When received from NWC the samples ranged in value from -8.0 to 1.0. To meet the assumption that the data exist in a lattice pattern space it

Figure 6.6 Preprocessing flowchart



was scaled into an integer space ranging from 0 to 300 in each component. Each real number component was scaled with the following procedure :

1. add 8.0
2. multiple by 32.0
3. add 0.5
4. truncate the fractional part.

Appendix F contains the resulting samples and the iterative processing results. In the first case of this study each class contained 16 samples.

The ten space data had a (I_{10}) ratio of 3.750768. The sufficient (I_2) ratio was .189089 when processing ceased. Twelve iterations were required to achieve this result. Figure 6.7a is the two space result of the transformation.

The iterative processing technique proved effective. After identifying the "best" value for each component of the reference vector in iterations 1 - 11, the "best" value for each component was placed into a single vector. This step yielded a result of 0.442269 and figure 6.7b. In examining each of the components of the mean sample for each of the classes it was observed that the components with the most variability between classes developed I_q minimizing reference point components which were not near the user defined bounds of plus or minus (9999,9999,...,9999). In this study components one, two, and three of the sample mean vectors exhibited this behavior. The remaining components

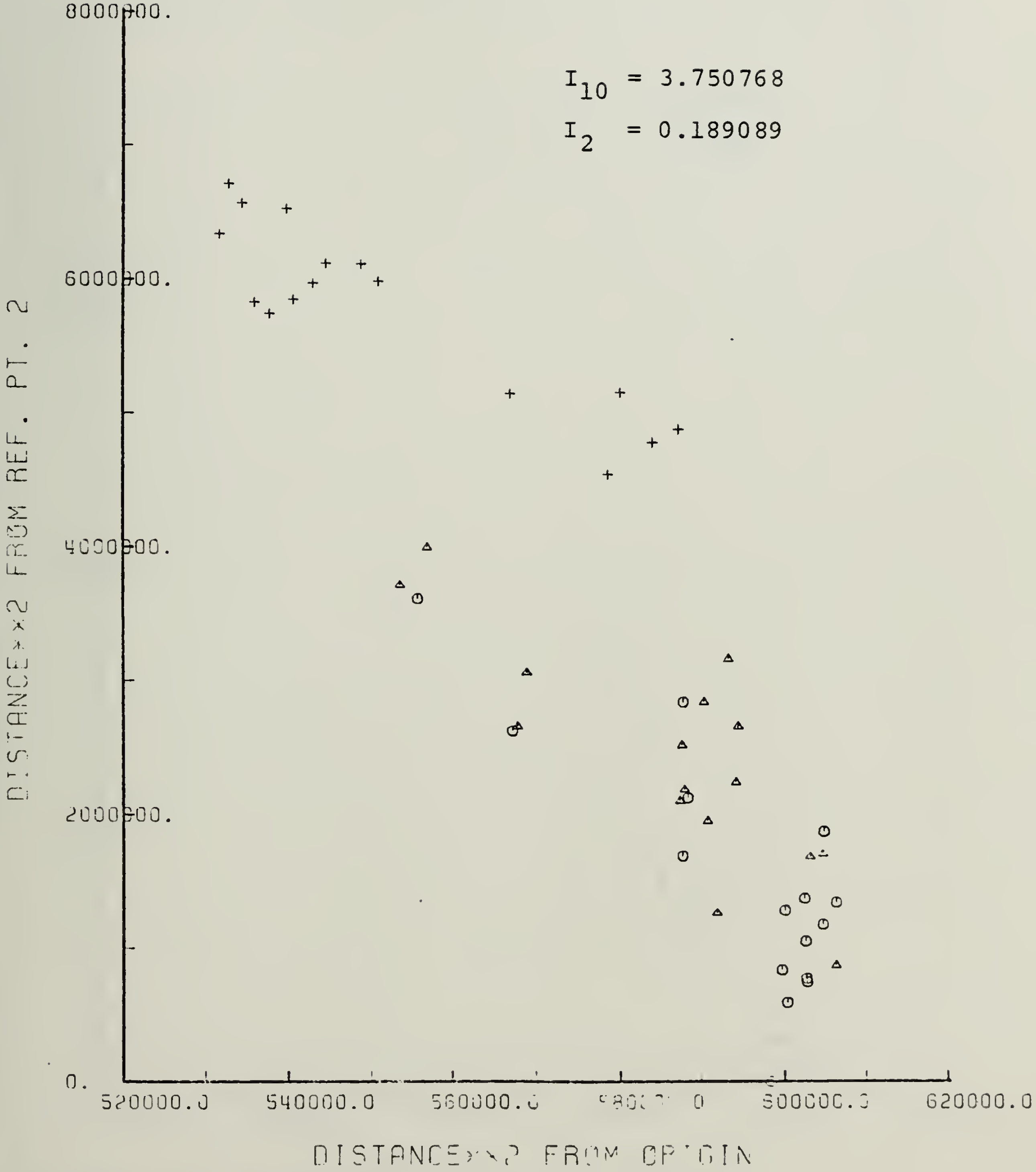


Figure 6.7a Ten space ship data 48 samples

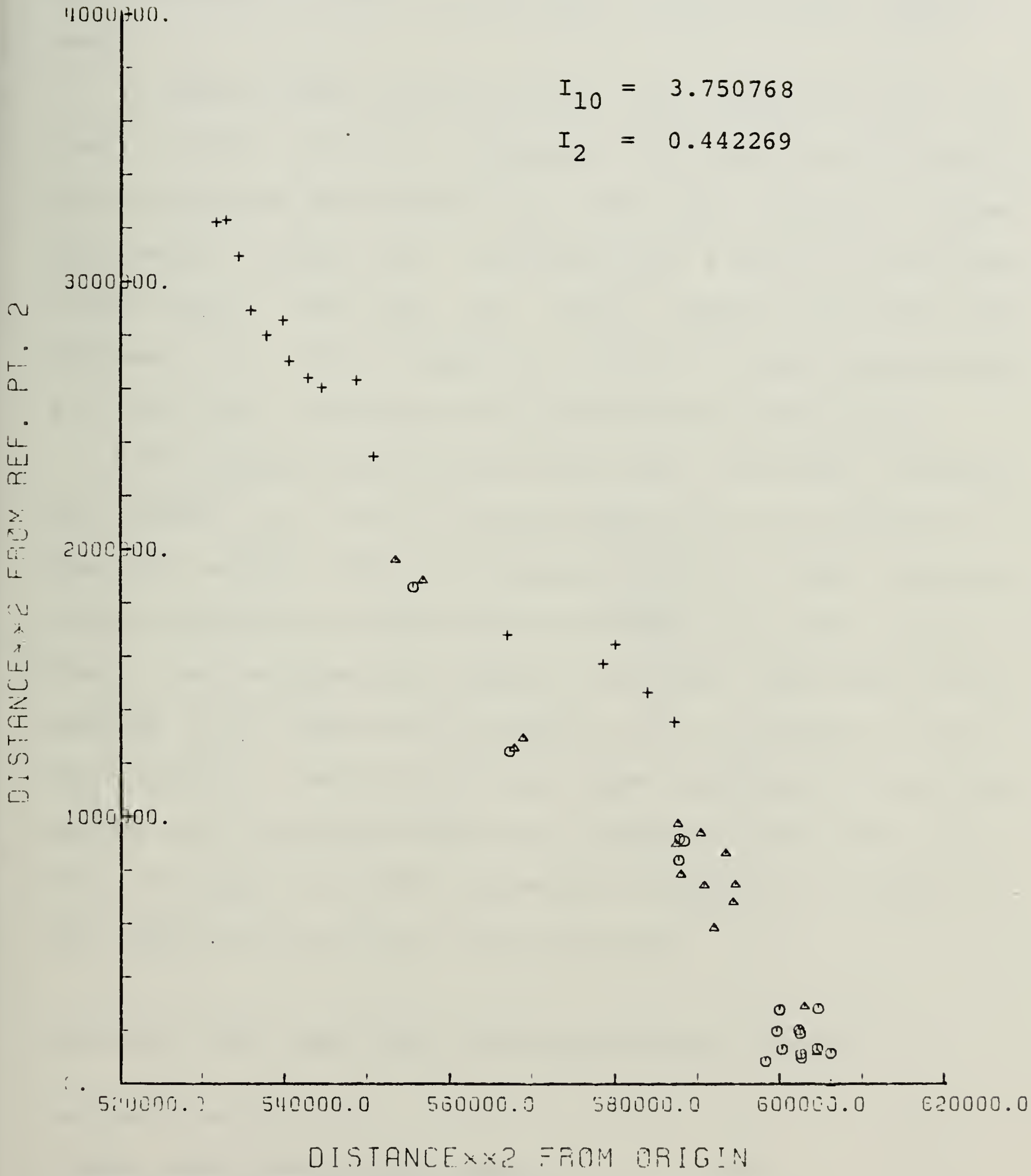


Figure 6.7b Ten space ship data 48 samples

had miminal variation of mean component values between classes. The remaining components generated reference point two component values at the user defined bounds, -9999 or 9999.

In making these observations it was decided that the "best" value for those components with values at the bound had not reached the minimum (I_2) for that element. These particular values were increased by a factor of ten and tested again. This was the vector which generated the minimum (I_2). This behavior is similar to that documented for the X and Y components of the bisected cube problem.

After determining the reference point two which yielded the lowest (I_2) over the 48 sample set it was decided to test the entire set of training samples. The training samples consisted of 423 samples divided into three classes. Class one contained 141 samples. Class two consisted of 137 samples. Class three had 145 samples. Two reference points were tested. The first was the value determined for the 48 sample set. The second point was similar to the first but with the negative -9999 components increased by a factor of ten. The following results were obtained.

Table 6.7 Ten space ship profiles reference points

reference point	I_2	figure
(-3000,-4500,-7000,-9999,-9999, 99999,99999,99999,99999,99999)	1.578162	6.7c

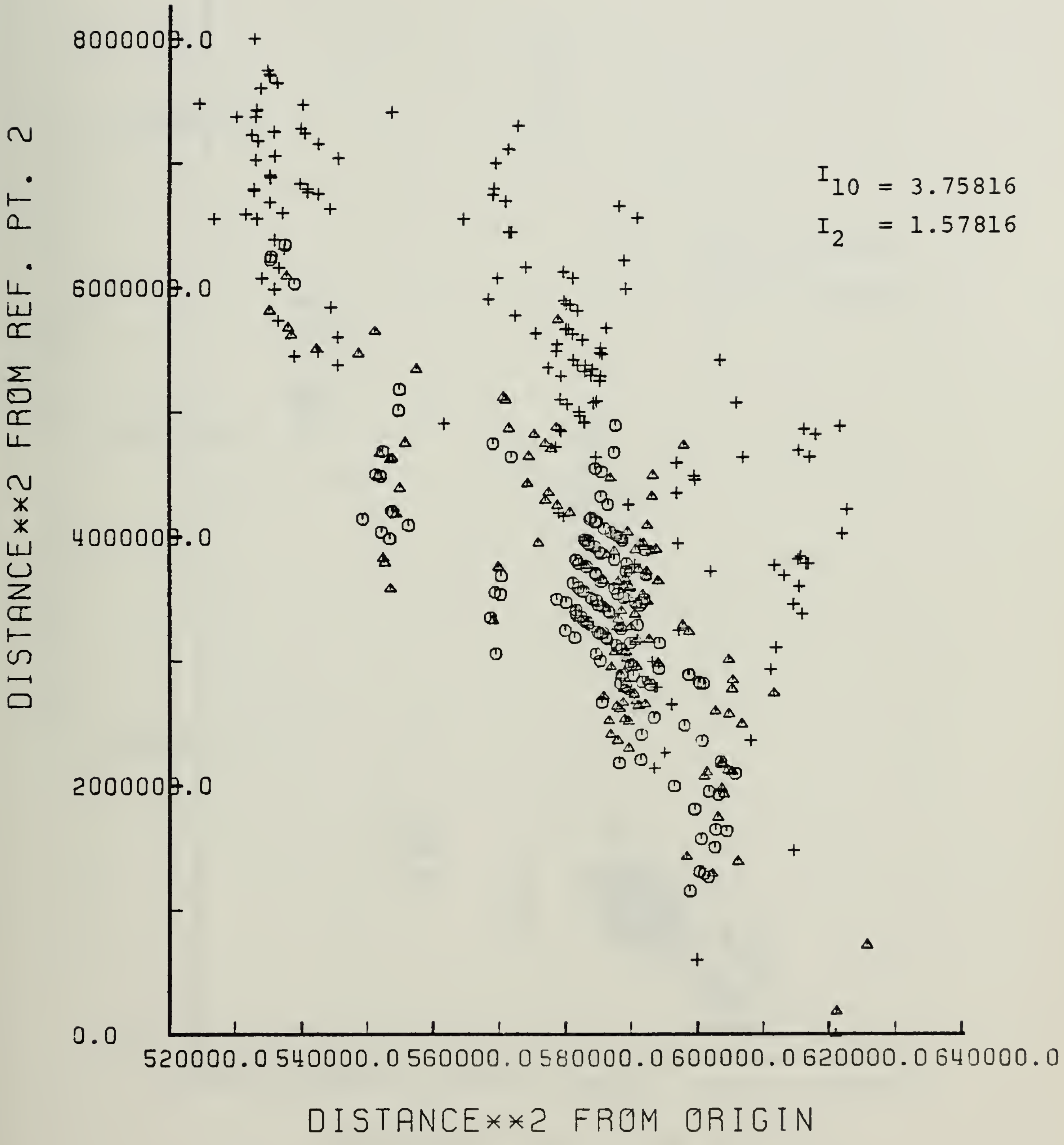


Figure 6.7c Ten space ship data 423 samples

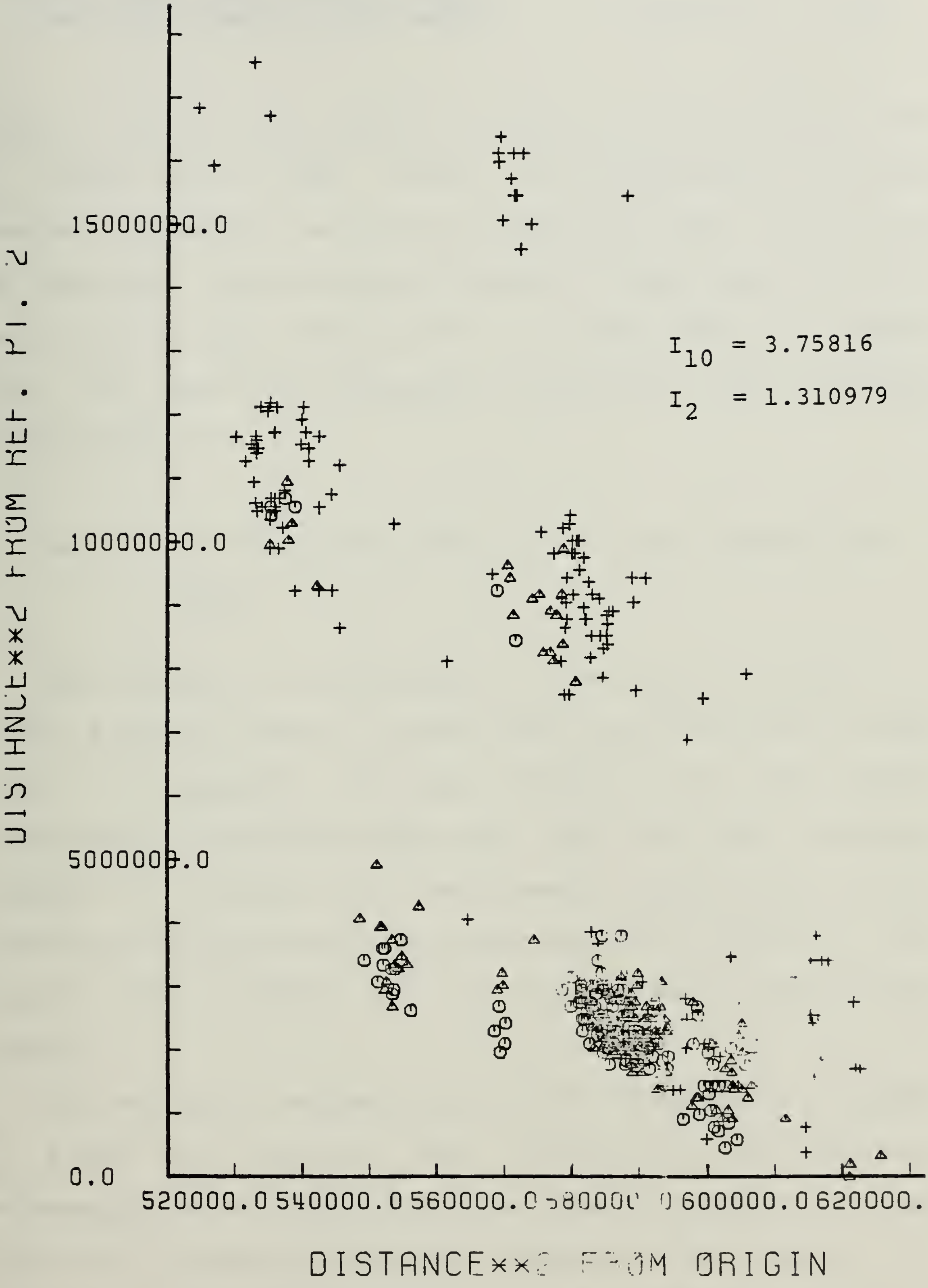


Figure 6.7d Ten space ship data 423 samples

(-3000,-4500,-7000,-99999,-99999, 99999,99999,99999,99999,99999)	1.310979	6.7d

Figure 6.7e is an enlargement of the heavily clustered area of figure 6.7c. While there is a large amount of noise present, as would be expected in real data, the claim will be made that three distinct clusters of data exist, one for each class of ship. Figure 6.7f is an enlargement of figure 6.7d. The same claim regarding clustering is made for this representation also.

G. CASE 7 : 32 SPACE THREE SHIP RADAR TARGET RECOGNITION DATA

This study is a continuation of the previous study in a higher dimension space. The same ship profiles were utilized with 32 components per sample instead of ten. The samples were scaled in the same manner as the ten space samples. Appendix G contains the references points tested in the iterative processing and the sample means for each of the classes. The samples are not listed due to their large number.

The two space representation of 32 dimensional samples in figure 6.8 exhibits much similarity to the ten space representation in figure 6.7c. The general location of the three ship classes is unchanged between the two figures. The reference point which generated the 0.189087 (I_2) ratio for

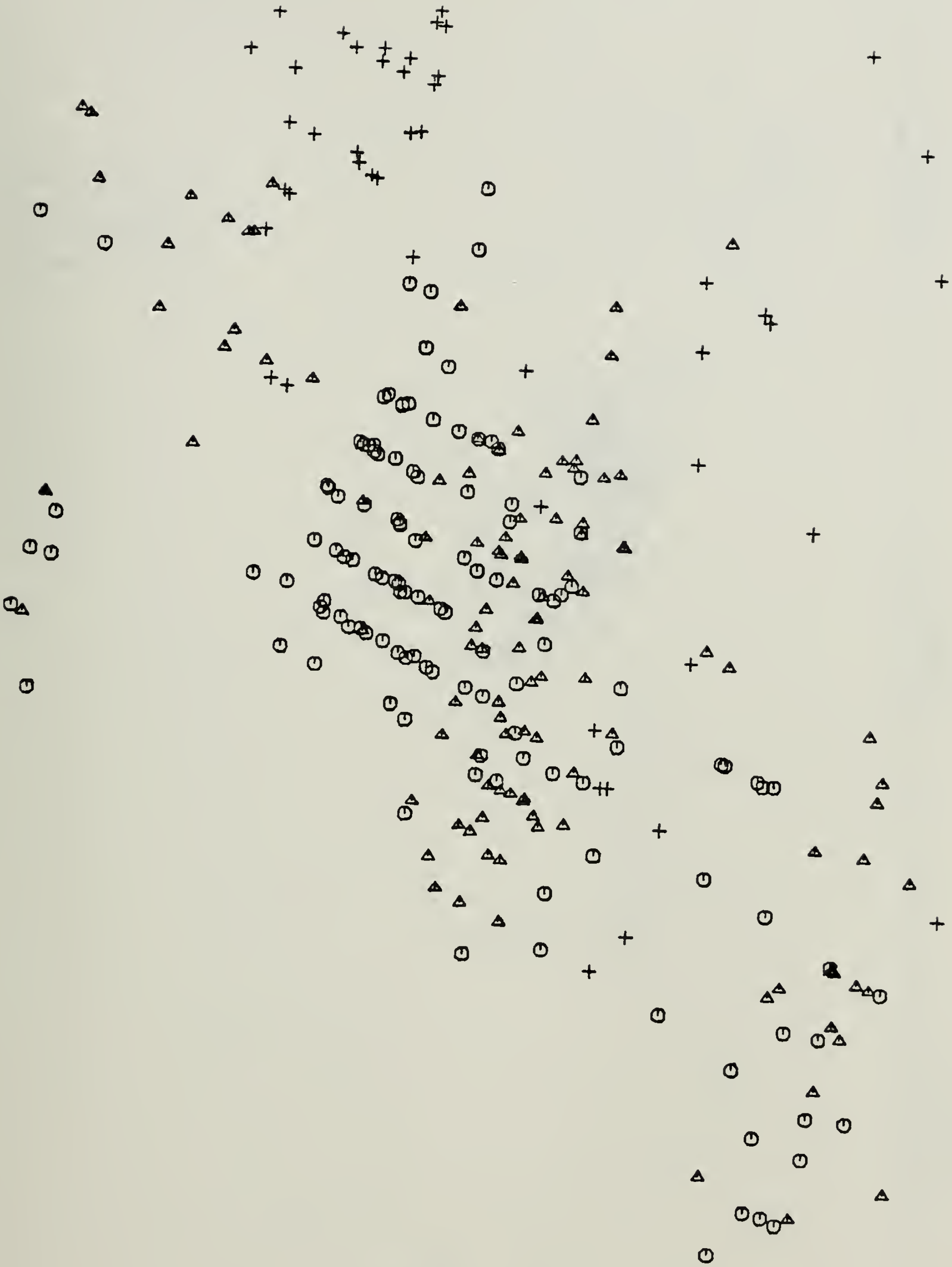
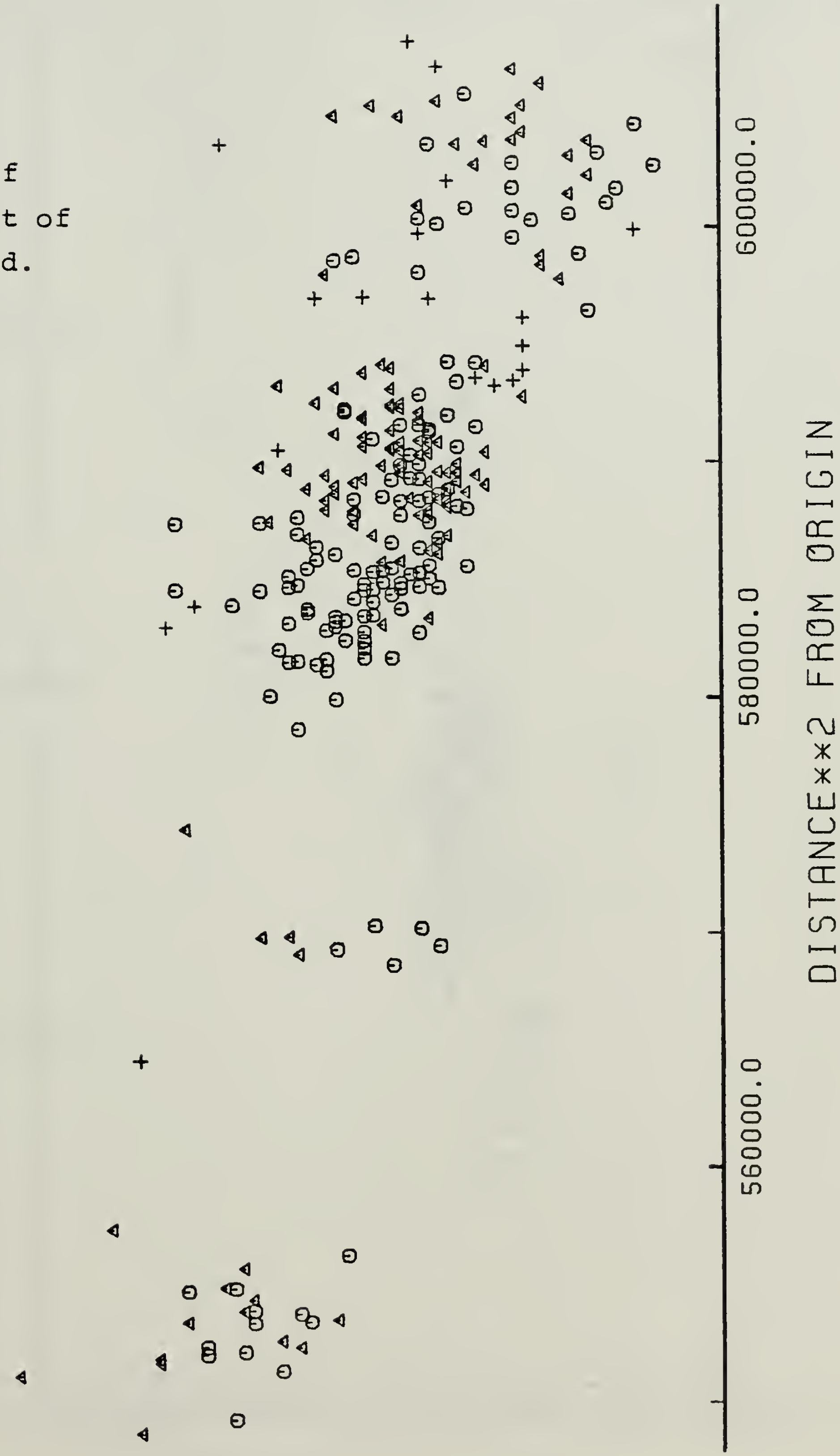


Figure 6.7e Enlargement of heavily clustered area of figure 6.7c.

Figure 6.7f
Enlargement of
figure 6.7d.



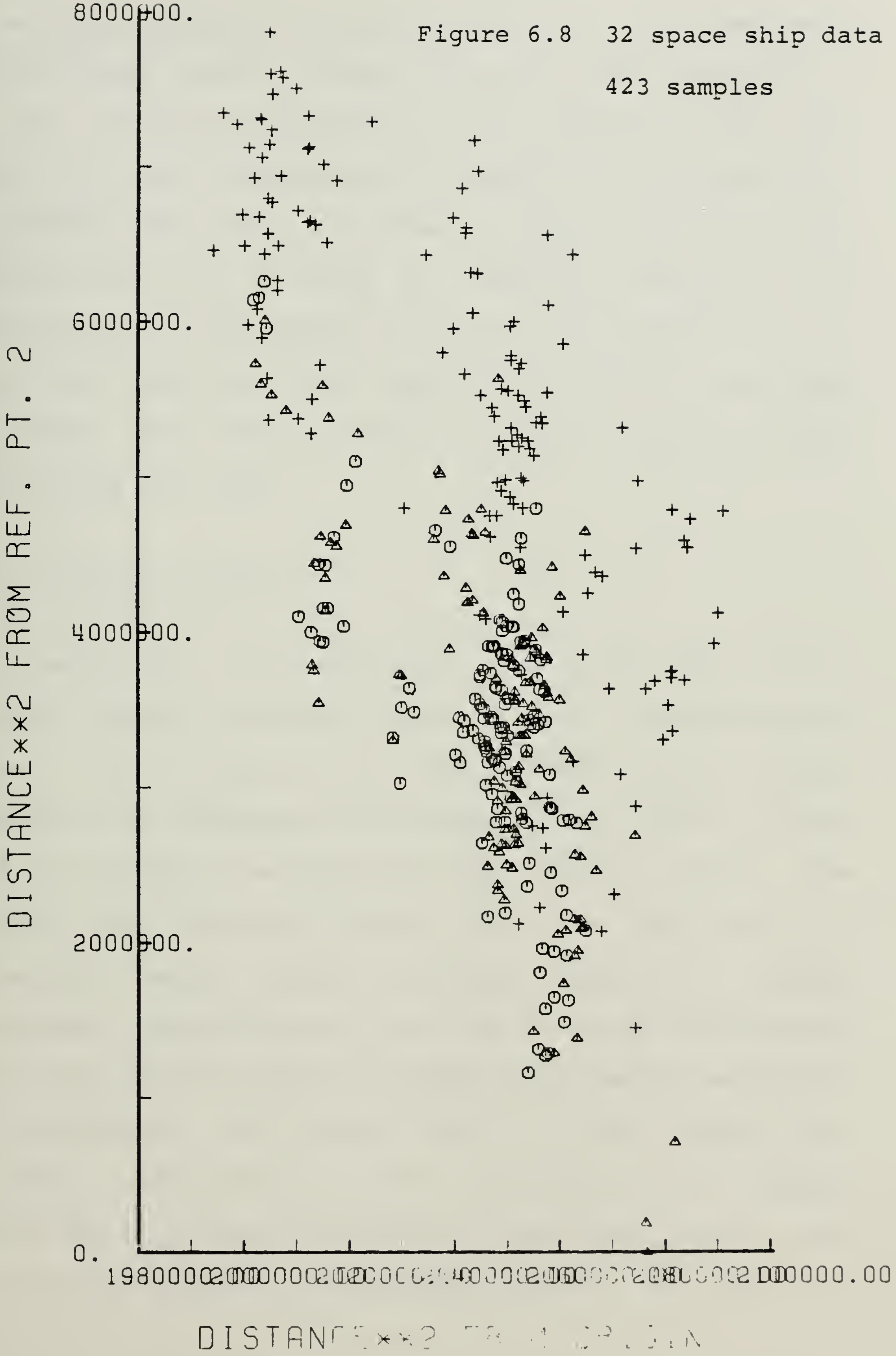


Figure 6.8 32 space ship data

423 samples

the ten space study was utilized as the first ten components of the 32 space vector. Sixteen iterations were required to test the first 24 components of the reference vector. The minimum (I_2) ratio obtained was 1.624708. The 32 space (I_{32}) was 3.607182. The similarity between the 32 and 10 space representations is as much as result of the data as the transformation. A discussion of the data and its meaning is beyond the scope of this thesis. The claim is again made that the two space representation contains three distinct clusters, one per ship.

H. CASE STUDIES SUMMARY

A series of seven cases have been tested. The studies highlight various effects found in the transformation process. The simple three class three space problem illustrated the changing relationship of data within classes when R_1 is fixed at the origin and R_2 is moved around. The bisected cube provided insight into how the curve of intersection passed through pattern space. It further demonstrated geometrically how the curve of intersection should cross pattern space to achieve the greatest amount of class clustering. The cube within a cube example was utilized to study the use of the (I_q) measure in a complex problem. The (I_q) ratio was shown to yield good results when used as an optimization criterion. However, it was also

shown that when different goals are desired (graphical display) (I) may not be the best method of achieving that objective. The three and ten space converging class studies demonstrated that as long as some separation existed between classes that separation could be enhanced in the lower dimensional representation. They further illustrated that the transformation will not remove class interleaving inherent in pattern space representations. Stated differently, the general relationships which exist in pattern space will be retained in the distance space representation. These two cases demomstrated that the transformation process can function effectively in a noisy environment.

VII. CONCLUSIONS

A nonlinear transformation has been developed which retains or improves relative proximity of similar information while significantly reducing the representations of data for several specific cases. These studies have shown the feasibility of this technique. They support the supposition that this transformation is a valid process suitable for general applications of dimensionality reduction.

The concept of information context is valid. By representing in m dimensional space the relative relationships of data in n dimensional space, rather than the data itself, there is essentially no loss of contextual information in the lower dimensional representation.

A. RECOMMENDATIONS FOR CONTINUED RESEARCH

1. First and foremost is testing the reduced representations in a pattern classification algorithm. To truly judge this an effective technique the pattern classification results must yield equal or better results than the classification results obtained with the n dimensional data.

2. Development of a better method of efficiently and

rapidly locating "optimal" reference points is critically needed. The current methods will become unworkable as higher dimension spaces are tested. The method could be some closed form analytically derivable result or an iterative search computational method.

3. High dimensions should be explored to observe any effects which may present themselves.

4. The minimum bound on the lattice point interval should be determined. This researcher feels it will be the smallest interval which will still uniquely represent information at the desired level of accuracy. As an example for single precision FORTRAN on the IBM 360 (7.5 digits accuracy) if the interval is 0.0001 then all points in pattern space must be capable of being uniquely represented within the remaining 3.5 digits of accuracy. A second bound on the lattice interval will be the discretization process. If an analog-to-digital converter collects eight bits of information then those bits define the information cells of the lattice.

5. Different measures of information context needed to be derived and compared to the current definition. Is the concept of measuring localized distance as defined by Meisel [2] a valid measure? Duda and Hart [1] also suggest several alternative measures.

6. The effect of assigning to reference point one values other than the origin should be studied.

7. Is there any significant advantage to mapping through a series of distance spaces to further enhance relative proximity? This could be done by one mapping to, for example, two space followed by several two space to two space transformations. A second method would be to map from n space to n - 1 space to n - 2 space to ... to two space.

8. Given that the data can be uniquely mapped into one dimension, can a reverse transformation be found to return the data to n dimensions? This question is complicated by the fact that the solution exist in noncontinuous space.

APPENDIX A TEST DATA FROM CASE STUDY 1

A. SAMPLE DATA

Sample	Class	Index number
(1, 1, 1)	1	1
(1, 3, 2)	1	2
(2, 1, 3)	1	3
(3, 2, 1)	1	4
(4, 4, 4)	2	5
(3, 4, 4)	2	6
(6, 5, 4)	3	7
(4, 5, 6)	3	8
(6, 4, 5)	3	9
(5, 4, 6)	3	10

B. REFERENCE POINT TEST

REFERENCE POINT	I_2
1. ITERATION 1	
(- 1, 1, 6)	.0966678
(6, 1, - 1)	.0946914
(- 1, 21, 8)	.2123157
(6, - 1, 1)	.0800436
(1, - 1, 6)	.0798612

APPENDIX B TEST DATA FROM CASE STUDY 2

A. SAMPLE DATA

Sample	Class
(1, 2, 0)	1
(1, 2, 1)	1
(1, 2, 2)	1
(1, 2, 3)	1
(1, 3, 0)	1
(1, 3, 1)	1
(1, 3, 2)	1
(1, 3, 3)	1
(1, 4, 0)	1
(1, 4, 1)	1
(1, 4, 2)	1
(1, 4, 3)	1
(2, 4, 0)	1
(2, 4, 1)	1
(2, 4, 2)	1
(2, 4, 3)	1
(2, 3, 0)	1
(2, 3, 1)	1
(2, 3, 2)	1
(2, 3, 3)	1
(3, 4, 0)	1
(3, 4, 1)	1
(3, 4, 2)	1
(3, 4, 3)	1
(2, 1, 0)	2
(2, 1, 1)	2
(2, 1, 2)	2
(2, 1, 3)	2
(3, 1, 0)	2
(3, 1, 1)	2
(3, 1, 2)	2
(3, 1, 3)	2
(3, 2, 0)	2
(3, 2, 1)	2
(3, 2, 2)	2
(3, 2, 3)	2
(4, 3, 0)	2
(4, 3, 1)	2

(4, 3, 2)	2
(4, 3, 3)	2
(4, 2, 0)	2
(4, 2, 1)	2
(4, 2, 2)	2
(4, 2, 3)	2
(4, 1, 0)	2
(4, 1, 1)	2
(4, 1, 2)	2

B. REFERENCE POINT TEST

REFERENCE POINT	I2

1. ITERATION 1

REFERENCE POINT	I2
(-1, 1, 6)	18.244934
(6, -1, 1)	1.575197
(-1, 21, 8)	.678884
(6, -1, -1)	.632244
(1, -1, 6)	18.244934

2. ITERATION 2

REFERENCE POINT	I2
(21, -1, 18)	1.534258
(-1, 18, 21)	2.392746
(21, 18, -1)	80.819992
(18, 21, -1)	80.819992
(18, -1, 21)	2.392746
(-1, 12, 15)	2.391603
(-1, 15, 12)	1.246020
(12, -1, 15)	2.391603
(15, -1, 12)	1.266020
(12, 15, -1)	36.019989
(15, 12, -1)	36.019989
(8, -1, 1)	.491111
(10, -1, 1)	.454380
(14, -1, 1)	.464800
(20, -1, 1)	.514694

3. Iteration 3

REFERENCE POINT	I2
(13, -1, 1)	.458060
(12, -1, 1)	.453136
(15, -1, 1)	.472577
(16, -1, 1)	.480899
(17, -1, 1)	.489444
(18, -1, 1)	.498007
(19, -1, 1)	.506450

(16, - 1, 2)	.480899
(18, - 1, 2)	.498007
(19, - 1, 2)	.506450
(20, - 1, 2)	.514694
(21, - 1, 2)	.522686
(12, - 1, 3)	.474437
(14, - 1, 3)	.480800
(16, - 1, 3)	.493355
(18, - 1, 3)	.507979
(20, - 1, 3)	.522857
(40, - 2, 6)	.602823
(40, 1, 6)	.748836

4. Iteration 4

(12, - 3, 1)	.336800
(12, - 4, 1)	.303828
(12, - 6, 1)	.267222
(12, - 12, 1)	.262812
(12, - 15, 1)	.281728
(12, - 20, 1)	.319706

5. Iteration 5

(12, - 13, 1)	.268448
(12, - 14, 1)	.274821
(12, - 16, 1)	.289005
(12, - 17, 1)	.296527
(12, - 90, 1)	.599325
(12, - 99, 1)	.613893

6. Iteration 6

(- 12, 12, 2)	.262812
(- 12, 12, 7)	.356562
(- 12, 12, 13)	.675312
(- 12, 12, 21)	1.450312
(- 12, 12, 27)	2.294062

7. Iteration 7

(- 12, 12, 3)	.269062
(- 12, 12, 5)	.300312
(- 50, 50, 3)	.203979
(50, - 50, 5)	.205779
(- 99, 99, 1)	.200923
(- 99, 99, 10)	.204229
(99, - 99, 10)	.207513

8. Iteration 8

(99, - 99, 1)	.200923
(150, - 150, 1)	.200402
(200, - 200, 1)	.200232
(- 150, 150, 1)	.200402
(- 200, 200, 1)	.200232
(- 99, 99, 3)	.200923
(- 50, 50, 1)	.201015
(50, - 50, 3)	.203619

9. Iteration 9

(300, - 300, 1)	.200100
(300, - 300, 3)	.200110
(- 300, 300, 5)	.200160
(- 300, 300, 30)	.204160

10. Iteration 10

(350, - 350, 1)	.200077
(- 400, 400, 1)	.200053
(- 500, 500, 1)	.200038
(- 500, 500, 100)	.217502
(- 400, 400, 100)	.227340

11. Iteration 11

(600, - 600, 1)	.200025
(- 700, 700, 3)	.200018
(800, - 800, 1)	.200191
(- 999, 999, 1)	.200012
(999, - 999, 3)	.200012

APPENDIX C TEST DATA FROM CASE STUDY 3

A. SAMPLE DATA

Sample	Class
(3, 3, 2)	1
(3, 3, 3)	1
(3, 2, 2)	1
(3, 2, 3)	1
(3, 4, 2)	1
(3, 4, 2)	1
(3, 4, 3)	1
(4, 3, 2)	1
(4, 3, 3)	1
(2, 3, 3)	1
(2, 3, 2)	1
(1, 1, 0)	2
(1, 2, 0)	2
(1, 3, 0)	2
(1, 4, 0)	2
(1, 5, 0)	2
(2, 1, 0)	2
(2, 2, 0)	2
(2, 3, 0)	2
(2, 4, 0)	2
(2, 5, 0)	2
(3, 1, 0)	2
(3, 2, 0)	2
(3, 3, 0)	2
(3, 4, 0)	2
(3, 5, 0)	2
(4, 1, 0)	2
(4, 2, 0)	2
(4, 3, 0)	2
(4, 4, 0)	2
(4, 5, 0)	2
(5, 1, 0)	2
(5, 2, 0)	2
(5, 3, 0)	2
(5, 4, 0)	2
(5, 5, 0)	2
(1, 1, 5)	2
(1, 2, 5)	2

```
( 1, 3, 5 )                         2
( 1, 4, 5 )                         2
( 1, 5, 5 )                         2
( 2, 1, 5 )                         2
( 2, 2, 5 )                         2
( 2, 3, 5 )                         2
( 2, 4, 5 )                         2
( 2, 5, 5 )                         2
( 3, 1, 5 )                         2
( 3, 2, 5 )                         2
( 3, 3, 5 )                         2
( 3, 4, 5 )                         2
( 3, 5, 5 )                         2
( 4, 1, 5 )                         2
( 4, 2, 5 )                         2
( 4, 3, 5 )                         2
( 4, 4, 5 )                         2
( 4, 5, 5 )                         2
( 5, 1, 5 )                         2
( 5, 2, 5 )                         2
( 5, 3, 5 )                         2
( 5, 4, 5 )                         2
( 5, 5, 5 )                         2
( 0, 1, 1 )                         2
( 0, 2, 1 )                         2
( 0, 3, 1 )                         2
( 0, 4, 1 )                         2
( 0, 5, 1 )                         2
( 0, 1, 2 )                         2
( 0, 2, 2 )                         2
( 0, 3, 2 )                         2
( 0, 4, 2 )                         2
( 0, 5, 2 )                         2
( 0, 1, 3 )                         2
( 0, 2, 3 )                         2
( 0, 3, 3 )                         2
( 0, 4, 3 )                         2
( 0, 5, 3 )                         2
( 0, 1, 4 )                         2
( 0, 2, 4 )                         2
( 0, 3, 4 )                         2
( 0, 4, 4 )                         2
( 0, 5, 4 )                         2
( 5, 1, 1 )                         2
( 5, 2, 1 )                         2
( 5, 3, 1 )                         2
( 5, 4, 1 )                         2
( 5, 5, 1 )                         2
( 5, 1, 2 )                         2
( 5, 2, 2 )                         2
( 5, 3, 2 )                         2
```

```
( 5, 4, 2 )                 2
( 5, 5, 2 )                 2
( 5, 1, 3 )                 2
( 5, 2, 3 )                 2
( 5, 3, 3 )                 2
( 5, 4, 3 )                 2
( 5, 5, 3 )                 2
( 5, 1, 4 )                 2
( 5, 2, 4 )                 2
( 5, 3, 4 )                 2
( 5, 4, 4 )                 2
( 5, 5, 4 )                 2
( 1, 5, 4 )                 2
( 2, 5, 4 )                 2
( 3, 5, 4 )                 2
( 4, 5, 4 )                 2
( 5, 5, 4 )                 2
( 1, 0, 4 )                 2
( 2, 0, 4 )                 2
( 3, 0, 4 )                 2
( 4, 0, 4 )                 2
( 5, 0, 4 )                 2
( 1, 0, 1 )                 2
( 2, 0, 1 )                 2
( 3, 0, 1 )                 2
( 4, 0, 1 )                 2
( 5, 0, 1 )                 2
( 1, 0, 2 )                 2
( 2, 0, 2 )                 2
( 3, 0, 2 )                 2
( 4, 0, 2 )                 2
( 5, 0, 2 )                 2
( 1, 5, 1 )                 2
( 2, 5, 1 )                 2
( 3, 5, 1 )                 2
( 4, 5, 1 )                 2
( 5, 5, 1 )                 2
( 1, 5, 2 )                 2
( 2, 5, 2 )                 2
( 3, 5, 2 )                 2
( 4, 5, 2 )                 2
( 5, 5, 2 )                 2
( 1, 0, 3 )                 2
( 2, 0, 3 )                 2
( 3, 0, 3 )                 2
( 4, 0, 3 )                 2
( 5, 0, 3 )                 2
( 1, 5, 3 )                 2
( 2, 5, 3 )                 2
( 3, 5, 3 )                 2
( 4, 5, 3 )                 2
```

(5, 5, 3)   2

B. REFERENCE POINT TEST

1. ITERATION 1

REFERENCE POINT	I_2
(21, - 01, 18)	506.106934
(- 1, 18, 21)	560.919434
(21, 18, - 1)	274.874023
(18, 21, - 1)	274.874023
(18, - 1, 21)	560.919434

2. ITERATION 2

REFERENCE POINT	I_2
(13, - 1, 1)	165.419464
(12, - 1, 1)	148.707962
(15, - 1, 1)	200.934296
(16, - 1, 1)	219.476013
(17, - 1, 1)	238.391846
(18, - 1, 1)	257.587891
(19, - 1, 1)	276.977295
(16, - 1, 2)	217.550034
(18, - 1, 2)	255.803177
(19, - 1, 2)	275.258789
(20, - 1, 2)	294.842041
(21, - 1, 2)	314.477783
(12, - 1, 3)	146.451431
(14, - 1, 3)	180.792053
(16, - 1, 3)	217.550156
(18, - 1, 3)	255.803207
(20, - 1, 3)	294.842041
(40, - 2, 6)	686.030518
(40, 1, 6)	621.195801

3. ITERATION 3

REFERENCE POINT	I_2
(- 1, 12, 15)	322.457764
(- 1, 15, 12)	289.038574
(12, - 1, 15)	322.457520
(15, - 1, 12)	289.038086
(12, 15, - 1)	178.622330
(15, 12, - 1)	178.622330
(8, 12, - 1)	92.785416
(10, - 1, 1)	118.168716
(14, - 1, 1)	182.874771
(20, - 1, 1)	296.498291
(- 1, 1, 6)	95.093445

```
(     6,     1, -  1 )        66.218964
( -   1,    21,    8 )       338.431396
(     6, -   1,    1 )        74.488647
(     1, -   1,    6 )        95.093933
```

4. ITERATION 4

```
(     6,     1,    6 )       146.091446
(    20,     6, -  5 )       263,564941
( -   7,     4,    6 )       281.805176
( -   6,    20,    6 )       435.876953
( -   6,    11,    6 )       275.054688
(     6,    14, -  7 )       206.950363
(     1, -   8,    6 )       388.758789
( -  10,    17,    6 )       567.541748
```

5. ITERATION 5

```
(    36,     6, -  1 )       487.041504
(   360,    60, -  1 )      1335.837891
(    64, -   8,    1 )      1160.075684
( - 500,   500,    6 )   1789688.000000
(    36, -   6,    1 )       731.306641
(   360, -  60,    1 )      2534.004395
(   640, -  80,    1 )      2574.547363
(    80,     8, -  1 )       942.906250
```

APPENDIX D TEST DATA FROM CASE STUDY 4

A. SAMPLE DATA

1. CASE 1

Class 1	Class 2
(7, 4, 4)	(16, 19, 21)
(5, 3, 2)	(20, 21, 19)
(2, 3, 4)	(22, 19, 17)
(5, 2, 4)	(22, 19, 18)
(6, 6, 2)	(20, 20, 24)
(4, 5, 0)	(17, 19, 18)
(4, 3, 2)	(22, 22, 19)
(6, 3, 2)	(21, 21, 18)
(5, 2, 5)	(26, 21, 21)
(0, 5, 4)	(19, 21, 17)
(4, 3, 1)	(21, 15, 20)
(3, 2, 8)	(18, 19, 21)

2. CASE 2

Class 1	Class 2
(9, 6, 5)	(14, 17, 19)
(7, 5, 4)	(18, 19, 17)
(4, 5, 6)	(20, 17, 15)
(7, 4, 6)	(20, 17, 16)
(8, 8, 4)	(18, 18, 24)
(6, 7, 2)	(15, 17, 16)
(6, 5, 4)	(18, 20, 17)
(8, 5, 4)	(19, 19, 16)
(7, 4, 7)	(24, 19, 19)
(0, 7, 6)	(17, 19, 15)
(6, 5, 3)	(19, 21, 16)
(5, 4, 10)	(18, 19, 22)

3. CASE 3

Class 1	Class 2
(11, 8, 7)	(12, 15, 17)
(9, 7, 6)	(16, 17, 15)
(6, 8, 8)	(18, 15, 13)
(9, 6, 8)	(18, 15, 14)
(10,10, 6)	(16, 16, 22)
(8, 9, 4)	(13, 15, 14)
(8, 7, 6)	(16, 18, 15)
(10, 7, 6)	(17, 17, 14)
(9, 6, 9)	(22, 17, 17)
(1, 9, 8)	(15, 17, 13)
(8, 7, 5)	(17, 19, 14)
(7, 6,12)	(16, 17, 20)

4. CASE 4

Class 1	Class 2
(13, 10, 9)	(10, 13, 15)
(11, 9, 8)	(14, 15, 13)
(8, 9, 10)	(16, 13, 11)
(11, 8, 10)	(16, 13, 12)
(12, 12, 8)	(14, 14, 20)
(10, 11, 6)	(11, 13, 12)
(10, 9, 8)	(14, 16, 13)
(12, 9, 8)	(15, 15, 12)
(11, 8, 11)	(20, 15, 15)
(3, 11, 10)	(13, 15, 11)
(10, 9, 7)	(15, 17, 12)
(9, 8, 14)	(14, 15, 18)

5. CASE 5

Class 1	Class 2
(15, 12, 11)	(8, 11, 13)
(13, 11, 10)	(12, 13, 11)
(10, 11, 12)	(14, 11, 9)
(13, 10, 12)	(14, 11, 10)
(14, 14, 10)	(12, 12, 18)
(12, 13, 8)	(9, 11, 10)
(12, 11, 10)	(12, 14, 11)
(14, 11, 10)	(13, 13, 10)
(13, 10, 13)	(18, 13, 13)
(6, 13, 12)	(11, 13, 9)
(12, 11, 9)	(13, 15, 10)
(11, 10, 16)	(12, 13, 16)

B. REFERENCE POINT TEST

REFERENCE POINT	I_2
1. CASE 1	
a. Iteration 1	
(-1, 42, 1)	0.093977
(42, -1, 1)	.106397
(-1, 57, 42)	.029866
(-1, 47, 125)	.045806
(77, 77, 1)	.034925
(77, 77, 77)	.014635
(127, -127, 3)	.029114
(-1, -6, 1)	.037278
(-73, -6, -1)	.046832
b. Iteration 2	
(999, 999, 999)	.016088
(999, 999, 1)	.029569
(500, 500, 500)	.015915
(500, 500, 1)	.029797
(500, 500, 999)	.020086
(500, 509, -999)	25.971329
c. Iteration 3	
(125, 125, 125)	.015041
(250, 250, 250)	.015589
(-1, 100, 100)	.023590
(500, 500, -1)	.029975
(-999, -999, 999)	.234499
(-999, -999, -999)	.016451
(500, -500, 500)	.203192
d. Iteration 4	
(55, 55, 55)	.014684
(67, 67, 67)	.014588
(73, 73, 73)	.014609
(33, 33, 33)	.017653
(27, 27, 27)	.021412
(49, 49, 49)	.014898
e. Iteration 5	
(66, 66, 66)	.014587
(63, 63, 63)	.014592
(69, 69, 69)	.014592
(61, 61, 61)	.014602
(59, 59, 59)	.014620

```
(     57,     57,     57 )                    .014646
```

2. CASE 2

a. Iteration 1

```
( -    1,     42,      1 )                    .127064
(     42, -    1,      1 )                    .186043
( -    1,     57,     42 )                    .043038
( -    1,     47,    125 )                    .088665
(     77,     77,      1 )                    .058361
(     77,     77,     77 )                    .027151
( -  127, -  127,      3 )                    .045052
( -    1, -    6,      1 )                    .051349
( -   73, -    6, -    1 )                    .080489
```

b. Iteration 2

```
(    999,    999,    999 )                    .028986
(    999,    999,      1 )                    .047432
(    500,    500,    500 )                    .028773
(    500,    500,      1 )                    .047993
(    500,    500,    999 )                    .038980
(    500,    500,   -999 )                 109.819839
```

c. Iteration 3

```
(     55,     55,     55 )                    .027149
(     67,     67,     67 )                    .027073
(     73,     73,     73 )                    .027113
(     33,     33,     33 )                    .030525
(     27,     27,     27 )                    .034905
(     49,     49,     49 )                    .027375
```

d. Iteration 4

```
(      66,      66,    66 )                   .027069
(      63,      63,    63 )                   .027066
(      69,      69,    69 )                   .027083
(      61,      61,    61 )                   .027073
(      59,      59,    59 )                   .027087
(      57,      57,    57 )                   .027112
```

3. CASE 3

a. Iteration 1

```
( -     1,      42,     1 )                   .240905
```

(42, - 1, 1)	.405388
(- 1, 57, 42)	.091247
(- 1, 47, 125)	.192553
(77, 77, 1)	.129220
(77, 77, 77)	.061053
(- 127, - 127, 3)	.097261
(- 1, - 6, 1)	.090916
(- 73, - 6, - 1)	.177960

b. Iteration 2

(999, 999, 999)	.063691
(999, 999,-999)	.910851
(- 999, - 999,-999)	.064311
(- 1, - 153, 1)	.065192
(- 999, - 999,- 1)	.102021
(- 999, - 999, 1)	.102291
(- 30, - 180, 1)	.060995

c. Iteration 3

(- 150, - 900, 1)	.064361
(66, 66, 66)	.060896
(63, 63, 63)	.060875
(- 180, - 30, 1)	.207636
(- 75, - 450, 1)	.063134
(75, 450,- 1)	.071101
(30, 180,- 1)	.083405

d. Iteration 4

(- 30, - 183, 1)	.061042
(- 27, - 180, 1)	.061103
(- 27, - 183, 1)	.061164
(- 30, - 180,- 1)	.062224
(- 30, - 180,- 30)	.049189
(- 33, - 180,- 33)	.049178
(- 30, - 180, 1)	.060965
(- 30, - 177, 1)	.060952

e. Iteration 5

(- 150, - 900,-900)	.066459
(- 150, - 900,-150)	.048069
(- 33, - 180,- 99)	.049638
(- 99, - 540,- 99)	.047232
(180, 33, 33)	.209693
(900, 150, 900)	.120490
(- 160, - 960,- 40)	.058638
(- 10, - 60,- 2)	.059914
(- 15, - 90,- 4)	.057103

(- 120, - 20, 5)	.200673
(- 120, - 20,- 5)	.178624

4. CASE 4

a. Iteration 1

(- 1, 42, 1)	.712316
(42, - 1, 1)	1.297319
(- 1, 57, 42)	.287248
(- 1, 47, 125)	.656638
(77, 77, 1)	.445736
(77, 77, 77)	.214881
(- 127, - 127, 3)	.331711
(- 1, - 6, 1)	.257414
(- 73, - 6,- 1)	.618422

b. Iteration 2

(- 30, - 180, 1)	.194983
(999, 999, 999)	.220502
(- 999, - 999, 0)	.352891
(- 999, - 999, 1)	.353415
(- 999, - 999,- 1)	.352478
(999, 999, 0)	.363830
(- 150, - 900, 1)	.208534
(- 150, - 900,- 1)	.207464
(66, 66, 66)	.214384
(63, 63, 63)	.214271
(59, 59, 59)	.214157
(57, 57, 57)	.214122
(61, 61, 61)	.214208
(67, 67, 67)	.214425

c. Iteration 3

(- 40, - 240, 1)	.197993
(- 40, - 240, 10)	.182372
(- 40, - 240, 0)	.196409
(- 50, - 300, 0)	.198964
(- 50, - 300, 50)	.301546
(- 20, - 120, 1)	.191039
(- 20, - 120,- 5)	.178492
(- 30, - 190, 1)	.195347
(- 30, - 170, 0)	.192591

d. Iteration 4

(- 150, - 900,-900)	.219152

```
( -  150, - 900,-150 )             .156107
( -   33, - 180,- 99 )             .159531
( -   99, - 540,- 99 )             .153407
(    180,    33,  33 )             .761994
(    900,   150, 900 )             .430386
( -  160, - 960,- 40 )             .190221
( -   10, -  60,-  2 )             .182952
( -  120, -  20,   5 )             .708054
```

5. CASE 5

a. Iteration 1

```
( -    1,    42,   1 )           18.325668
(     42, -   1,   1 )           38.237442
( -    1,    57,  42 )            8.180267
( -    1,    47, 125 )           28.016891
(     77,    77,   1 )           21.455872
(     77,    77,  77 )           11.546129
( -  127, - 127,   3 )           15.333944
( -    1, -   6,   1 )            9.163347
( -   73, -   6,-  1 )           53.390564
```

b. Iteration 2

```
( -  150, - 900,   1 )            5.632728
( -   30, - 180,   1 )            5.433392
( -   75, - 450,   1 )            5.564706
(      0,  1011,7008 )           41.082291
(      1,    86,  63 )            7.130882
( -    1,   234, 168 )            6.409952
( -  150, - 900,   0 )            5.621341
```

c. Iteration 3

```
( -   40, - 240,   1 )            5.484715
( -   40, - 240,- 10 )            5.150225
( -   40, - 240,   0 )            5.450282
( -   50, - 300,   0 )            5.485142
( -   50, - 300,  50 )            7.647579
( -   20, - 120,   1 )            5.441672
( -   20, - 120,-  5 )            5.168211
( -   30, - 190,   1 )            5.413536
( -   30, - 170,   0 )            5.458088
```

d. Iteration 4

```
( -  150, - 900,-900 )            8.136481
( -  150, - 900,-150 )            4.560885
```

```
( -   33, - 180,- 99 )              5.401924
( -   99, - 540,- 99 )              4.591217
(    180,    33,  33 )             99.351578
(    900,   150, 900 )             38.806274
( -  160, - 960,- 40 )              5.247082
( -   10, -  60,-  2 )              5.492326
( -  120, -  20,   5 )             66.094223
( -  120, -  20,-  5 )             57.096405
```

APPENDIX E TEST DATA FROM CASE STUDY 5

A. SAMPLE DATA

1. CASE 1

```
        (    a,   b,   c,   d,   e,   f,   g,   h,   i,   j )
Class 1
        (    5,   3,   3,   4,   1,   4,   5,   5,   5,   1 )
        (    4,   4,   5,   4,   3,   6,   4,   6,   0,   2 )
        (    0,   5,   5,   4,   3,   6,   8,   2,   4,   3 )
        (    5,   4,   3,   5,   4,   2,   5,   6,   4,   3 )
        (    3,   8,   3,   4,   4,   6,   3,   5,   1,   4 )
        (    5,   4,   5,   4,   1,   7,   1,   0,   3,   2 )
        (    5,   4,   4,   4,   4,   4,   5,   3,   6,   2 )
        (    2,   2,   4,   7,   6,   4,   6,   2,   3,   6 )
        (    4,   7,   6,   3,   4,   5,   2,   5,   3,   5 )
        (    4,   3,   5,   4,   4,   4,   4,   5,   7,   4 )
        (    4,   5,   3,   7,   3,   4,   4,   4,   6,   3 )
        (    5,   2,   4,   6,   2,   1,   7,   4,   2,   6 )
        (    3,   0,   7,   4,   4,   5,   2,   4,   4,   4 )
        (    4,   4,   5,   5,   1,   4,   3,   4,   4,   4 )
        (    7,   4,   3,   4,   1,   4,   4,   4,   4,   7 )
        (    6,   2,   0,   4,   2,   6,   5,   2,   4,   6 )
```

```
Class 2
        (   16,  12,  18,  16,  15,  19,  16,  15,  16,  17 )
        (   17,  14,  19,  16,  16,  13,  15,  14,  16,  18 )
        (   14,  16,  15,  15,  19,  13,  14,  16,  19,  18 )
        (   16,  18,  15,  15,  15,  15,  17,  15,  17,  17 )
        (   16,  16,  11,  18,  15,  13,  14,  15,  16,  14 )
        (   13,  16,  15,  16,  19,  15,  15,  16,  20,  16 )
        (   12,  16,  16,  17,  16,  17,  19,  18,  16,  16 )
        (   16,  15,  15,  17,  16,  17,  16,  16,  16,  17 )
        (   16,  16,  14,  19,  16,  16,  16,  17,  18,  17 )
        (   17,  15,  17,  17,  16,  15,  16,  17,  18,  13 )
        (   17,  18,  17,  16,  15,  16,  19,  19,  16,  16 )
        (   14,  18,  12,  17,  13,  13,  17,  17,  15,  16 )
        (   16,  13,  16,  13,  16,  16,  18,  16,  16,  16 )
        (   18,  14,  15,  16,  14,  16,  16,  16,  19,  16 )
        (   16,  18,  16,  18,  17,  18,  15,  17,  18,  13 )
        (   13,  16,  16,  18,  17,  14,  16,  18,  16,  18 )
```

2. CASE 2

```
Class 1
          (   5,  7,  4,  6,  6,  8,  8,  5,  6,  6 )
          (   3,  2,  4,  8,  3,  5,  7,  3,  6,  8 )
          (   8,  4,  6,  5,  5,  5,  4,  5,  6,  5 )
          (   5,  7,  8,  7,  8,  5,  8,  6,  9, 11 )
          (   6,  8,  7,  4,  4,  2,  9, 11,  9,  9 )
          (   4, 10,  6,  4,  6,  6,  6,  6,  5,  8 )
          (   6,  6,  7,  6,  9,  6,  5,  6,  8,  6 )
          (   6,  3,  6,  8,  5,  6,  6,  5,  7,  4 )
          (   4,  6,  5,  8,  6,  6,  2,  6,  5,  6 )
          (   4,  7,  6,  6,  7,  6,  7,  6,  4,  5 )
          (   8,  6,  6,  5,  6,  8,  4,  6,  7,  8 )
          (   7,  7,  6,  9,  6,  5,  5,  6,  7,  6 )
          (   2,  7,  4,  5,  9,  6,  9,  6,  4,  6 )
          (   3,  6,  2,  3,  8,  5,  4,  6,  5,  6 )
          (   5,  3,  5,  6, 10,  8,  4,  5,  6,  4 )


Class 2
          (  17, 21, 14, 15, 15, 14, 14, 14, 15, 15 )
          (  16, 16, 14, 14, 14, 15, 15, 14, 14, 14 )
          (  14, 16, 13, 15, 19, 12, 17, 14, 17, 15 )
          (  11, 15, 14, 15, 15, 16, 14, 16, 14, 14 )
          (  14, 15, 14, 15, 15, 13, 15, 12, 12, 10 )
          (  13, 15, 12, 14, 14, 14, 13, 16, 14, 14 )
          (  15, 15, 14, 17, 15, 17, 12, 11, 11, 12 )
          (  14, 13, 13, 15, 14, 15, 17, 14, 14, 16 )
          (  14, 14, 12, 16, 15, 14, 15, 13, 13, 17 )
          (  18, 16, 14, 14, 13, 14, 13, 13, 14, 13 )
          (  14, 14, 12, 12, 14, 11, 12, 11, 14, 14 )
          (  13, 13, 13, 14, 14, 13, 15, 14, 14, 15 )
          (  17, 12, 14, 13, 14, 14, 12, 13, 16, 15 )
          (  16, 16, 14, 13, 16, 13, 14, 13, 11, 14 )
          (  13, 14, 14, 11, 13, 16, 16, 14, 14, 14 )
          (  14, 17, 11, 12, 13, 14, 14, 14, 14, 18 )
```

3. CASE 3

Class 1

```
(   8,  8,  8, 10,  7,  7,  8,  8,  7,  7 )
(  12,  7,  8,  8,  7,  9, 10,  8,  7,  3 )
(   7, 12,  8, 10,  7,  7,  8,  9,  9,  8 )
(  11,  8,  8,  8,  8, 10, 11,  8,  8,  6 )
(   8,  8,  8,  8, 10,  8,  7,  3,  6,  8 )
(  11, 10,  6,  8, 10,  8, 10, 10,  9,  9 )
(   9,  6,  8,  8,  8, 10,  9,  7,  5,  8 )
(   7,  7,  9,  6, 10,  7, 10,  8,  8,  9 )
(   9,  9, 10,  7,  7,  7.  6,  9,  8,  9 )
(   7,  9,  7,  7,  8,  6,  7, 10, 10, 10 )
(   8,  9,  8,  9, 10,  8, 14, 10,  6, 11 )
(   8, 11,  7,  8,  6,  8,  7,  9,  8,  9 )
(   9,  7,  8,  9,  7,  9,  6,  8,  8, 10 )
(   9,  8,  8,  7,  8, 10,  6,  4,  7,  7 )
(   9,  8, 10,  8,  6,  8,  8,  9,  6,  8 )
```

Class 2

```
(  12, 12, 10, 12, 12, 13, 12, 13, 12, 12 )
(  13, 13, 15, 12,  8, 13, 14,  9, 12, 14 )
(  13, 13, 13, 12, 12, 10, 14, 12, 11, 12 )
(  12, 10, 14, 16, 13, 16, 10, 10, 14, 13 )
(  11, 12,  8, 12, 12, 11, 11,  9, 11, 11 )
(  12, 12, 12, 13, 12, 14, 13, 12, 11, 13 )
(  12, 11, 12, 13, 15, 12,  8, 12, 11, 14 )
(  12, 12, 13, 12, 13, 14, 15, 11,  8, 15 )
(  14, 15, 11, 11, 11, 13, 10, 13, 12, 12 )
(  12, 13, 13, 13, 12, 11, 12, 12, 11. 13 )
(   8, 14, 14, 10, 11, 12, 11, 14, 11, 12 )
(  12, 13, 12, 12,  9, 12, 14, 12, 10, 10 )
(  12, 13,  9, 14, 12, 11, 14, 12, 11, 12 )
(  12, 13, 13, 12, 12, 14, 11, 12, 14, 12 )
(  12, 13, 14, 11, 12, 12, 12, 13, 15, 12 )
(  14, 13, 13, 14, 15,  9, 14, 16, 12, 11 )
```

4. CASE 4

Class 1

```
(   9, 10, 13, 11, 11, 11, 10, 12,  9, 10 )
(  10, 10, 10,  8, 14,  9, 10, 9,   9,  10 )
(  10,  9, 12,  9,  9, 10,  9,  9, 10, 10 )
(  12, 11, 17, 10, 11, 12, 15, 11, 10, 10 )
(  10,  7, 11, 12, 10, 10, 10,  9,  8, 10 )
(  10, 10, 10, 11,  9, 10, 11, 10,  9, 10 )
(  10,  9, 10,  9,  8, 13, 11, 10,  8,  9 )
(  11, 10,  8,  8,  7, 13,  7, 12, 10, 10 )
(  10, 10,  8, 10, 10,  8,  9,  8, 10, 10 )
(  11, 10, 10, 10, 10, 10, 12, 14, 10, 13 )
(  10, 10, 10, 12, 10, 12, 11, 11,  9,  6 )
(  11, 10, 15, 10, 12,  8, 10,  8, 11, 10 )
(   8, 11, 10, 11,  9, 12, 11,  9,  9, 10 )
(  10, 10, 10, 10, 12,  8, 10,  6, 10, 14 )
(   9,  8, 11, 13, 10,  9, 10, 10,  9, 11 )
(   6, 11, 11, 11,  7, 10, 10, 11, 13, 10 )
```

Class 2

```
(   9, 12, 12, 10,  7, 13,  8, 10, 10, 11 )
(  12,  8, 11, 10, 10, 10, 10, 10, 12, 10 )
(  12, 14,  9, 12, 10, 12,  6, 10, 10, 14 )
(  11, 10, 10, 11, 11, 11, 10,  9,  8,  6 )
(   8, 12, 10, 10, 13, 10, 10, 10, 10, 12 )
(   8,  8, 10, 13,  9, 10,  9,  9,  6, 12 )
(  12, 13,  9, 10, 12, 10,  9,  9, 13,  6 )
(  10,  6, 11, 10,  8, 11, 10,  7, 10, 10 )
(   8, 11,  8, 11, 11, 12, 12, 11,  9,  9 )
(  10,  9, 10,  9, 10, 12, 10,  8,  8,  8 )
(  10,  9, 11,  9, 10,  1,  1,  1,  1,  1 )
(  10,  9, 10, 11,  8,  9, 10, 10, 11, 10 )
(   9,  9,  9,  8, 10, 10, 12, 10,  9, 15 )
(  10, 10, 11,  8,  9,  7,  9, 11, 11, 10 )
(  12, 10, 12, 10, 10,  9, 10, 10, 10, 10 )
```

B. REFERENCE POINT TEST

1. CASE 1

a. Iteration 1

REFERENCE POINT	I_2
(51, 52, 53, 54, 55, 56, 57, 58, 59, 60)	.004561
(121, 171, 182, 175, 123, 245, 186, 263, 298, 500)	.005734
(- 57,- 58,- 59,- 60,- 61,- 62,- 63,- 64,- 65,- 66)	.005002
(-501,-502,-503,-504,-505,-506,-507,-508,-509.-510)	.004654
(0,-721,-821,-921,-111,-211,-311,-411,-511,-611)	.006724

2. CASE 2

a. Iteration 1

(51, 52, 53, 54, 55, 56, 57, 58, 59, 60)	.024579
(121, 171, 182, 175, 123, 245, 286, 263, 298, 500)	.034334
(- 57,- 58,- 59,- 60,- 61,- 62,- 63,- 64,- 65,- 66)	.023966
(-501,-502,-503,-504,-505,-506,-507,-508,-509,-510)	.023578
(0,-721,-821,-921,-111,-211,-311,-411,-511,-611)	.032695

3. CASE 3

a. Iteration 1

(51, 52, 53, 54, 55, 56, 57, 58, 59, 60)	.063348
(121, 171, 182, 175, 123, 245, 286, 263, 298, 500)	.078378
(- 57,- 58,- 59,- 60,- 61,- 62,- 63,- 64,- 65,- 66)	.063750
(-501,-502,-503,-504,-505,-506,-507,-508,-509,-510)	.062027
(0,-721,-821,-921,-111,-211,-311,-411,-511.-611)	.075525

4. CASE 4

a. Iteration 1

(51, 52, 53, 54, 55, 56, 57, 58, 59, 60)	20.958557
(121, 171, 182, 175, 123, 245, 286, 263, 298, 500)	23.204590
(-57, -58, -59, -60, -61, -62, -63, -64, -65, -66)	21.062271
(-501,-502,-503,-504,-505,-506,-507,-508,-509,-510)	21.051666
(0,-721,-821,-921,-111,-211,-311,-411,-511,-611)	17.263016

b. Iteration 2

```
(     0,-721,-821,-921,-111,-211,-311,-411,-511,  -1)  14.781326
(     0,  -1,-821,-921,-111,-211,-311,-411,-511,  -1)  13.008581
(     0,  -1,-821,-921,-111,-211,-311,-411,-511,-611)  12.028761
(     0,  -1,-821,-921,-111,-900,-311,-411,-511,-611)   9.707917
(     0,  -1,-821,-921,-111,-900,-311,-411,-511,-611)  15.126908
```

c. Iteration 3

```
(     0,  -1,-821,-921,-111,-900,   0,-411,-511,  -1)  19.077774
(     0,  -1,-821,-921,-111, 900, 999,-411,-511,  -1) 29116.8437
(     0,  -1,-821,-921,-111, 900,-311,-411,-511, 500)  18.246217
(     0,  -1,-821,-421,-111, 900,-311,-411,-511,-611)  13.567417
(     0,  -1,-221,-921,-111, 900,-311,-411,-511,-611)  51.881021
```

d. Iteration 4

```
(     0,  -1,-821,-999,-111, 999,-311,-411,-511,-611)  13.752581
(     0,-500,-821,-921,-111, 900,-311,-411,-511,-611)  15.657303
(     0,  -1,-821,-921,-111, 900,-999,-411,-511,-611)   9.724404
(     0,  -1,-821,-921,-111, 900,-666,-411,-511,-611)  10.710649
(  -500,  -1,-821,-921,-111, 900,-311,-411,-511,-611)  15.677959
```

e. Iteration 5

```
(  -500,  -1,-821,-219,-111, 900,-311,-411,-511,-611)  17.799042
(     0,  -1,-821,-219,-111, 900,-666,-411,-511,-611)  11.379161
(     0,  -1,-821,-219,-111, 900,-999,-411,-511,-611)  10.225633
(     0,   0,-821,-219,-111, 900,-999,-411,-511,-611)  10.217400
```

f. Iteration 6

```
(     0, 200,-821,-921,-111, 900,-999,-411,-511,-611)   9.311914
(     0, 200,-821,-219,-111, 900,-999,-411,-511,-611)   9.743191
(     0,  -1,-821,-921,-111, 900,-222,-411,-511,-611)  13.919784
(     0,  -1, 821,-921,-111, 900,-999,-411,-511,-611)   9.724404
(     0,  -1,-821,-921,-111, 900,-311,-411,-999,-611)  14.558681
(     0,  -1,-821,-921,-111, 900,-311,-411,   1,-611)  13.070512
(     0,  -1,-821,-921,-111, 900,-311,-411, 500,-611)  16.673279
(     0,  -1,-821,-219,-111, 900,-999,-411,-511,-611)  10.225633
```

g. Iteration 7

```
(     0, 800,-821,-921,-111, 900,-999,-411,-511,-611)   8.751494
(     0,  -1, 999,-921,-111, 900,-311,-411,-511,-611)  1364.4433
(     0,  -1, 999,-921,-111, 900,-999,-411,-511,-611)  59.601532
(     0, 800, 999,-921,-111, 900,-411,-411,-511,-611)  40.654962
(     0,  -1,-821,-921,-111, 900,-311,-411,-511,-999)  15.521662
(     0, 800, 999,-921,-111, 900,-999,-411,-511,-999)  43.365723
(     0, 800, 999,-921,-111, 900,-999,-999,-511,-999)  31.393661
```

```
(    0, 800, 999,-921,-111, 900,-999, 500,-511,-999) 158.365446
(    0, 800, 999,-921,-111, 900,-999, 999,-511,-999) 11577.5690
(    0, 800, 999,-921,-111, 900,-999,   1,-511,-999)  64.88529
(    0, 800, 999,-921,-111, 900,-999,-500,-511,-999)  40.597549
```

h. Iteration 8

```
(    0, 800,-821,-921,-111, 900,-999,-411,-511,-611)   8.751494
(  500, 800,-821,-219,-111, 900,-999,-411,-511,-611)   8.468982
(  500,  -1,-821,-921,-111, 900,-311,-411,-511,-611)  11.770139
(  999,  -1,-821,-921,-111, 900,-311,-411,-511,-611)  11.542337
(  999, 800,-821,-219,-111, 900,-999,-411,-511,-611)   8.294548
(    0, 800,-821,-216,-555, 900,-999,-411,-511,-611)   9.677239
(    0, 800,-821,-219, 555, 900,-999,-411,-511,-611)   9.246835
```

APPENDIX F TEST DATA FROM CASE STUDY 6

A. SAMPLE DATA

1. PHASE 1 : 48 SAMPLE TEST

```
        (    a,   b,   c,   d,   e,   f,   g,   h,   i,   j )

Class 1

        (    4, 124, 239, 267, 266, 263, 262, 260, 261, 260 )
        (   18, 230, 271, 264, 264, 261, 261, 259, 260, 260 )
        (   19, 230, 271, 265, 264, 262, 262, 259, 259, 260 )
        (    2, 225, 239, 266, 265, 262, 261, 260, 260, 260 )
        (  190, 101, 237, 268, 268, 263, 262, 261, 261, 260 )
        (    8, 227, 239, 266, 264, 262, 260, 260, 260, 259 )
        (   10, 227, 271, 265, 263, 262, 260, 260, 259, 259 )
        (    4, 225, 272, 266, 263, 263, 261, 260, 260, 260 )
        (   39, 233, 262, 264, 263, 263, 261, 260, 259, 259 )
        (   32, 231, 263, 265, 263, 259, 260, 259, 258, 259 )
        (   19, 227, 264, 266, 263, 261, 261, 259, 258, 259 )
        (   26, 229, 263, 266, 263, 260, 261, 260, 259, 259 )
        (   29, 230, 264, 266, 263, 260, 260, 260, 259, 259 )
        (   24, 230, 265, 268, 264, 260, 260, 261, 259, 259 )
        (   16, 227, 267, 269, 264, 260, 260, 261, 259, 259 )
        (    8, 124, 267, 268, 263, 262, 261, 261, 260, 259 )

Class 2

        (    1, 115, 239, 268, 265, 263, 262, 261, 260, 260 )
        (  191, 113, 240, 268, 263, 263, 261, 261, 259, 260 )
        (   19, 225, 240, 268, 265, 265, 262, 262, 259, 261 )
        (    6, 225, 269, 265, 265, 262, 262, 260, 259, 260 )
        (  191, 114, 271, 266, 266, 263, 262, 260, 259, 260 )
        (    2, 122, 270, 266, 264, 263, 262, 259, 259, 260 )
        (    6, 122, 239, 268, 266, 265, 262, 260, 260, 261 )
        (    2, 120, 271, 266, 264, 264, 261, 260, 260, 260 )
        (   19, 228, 234, 269, 263, 267, 259, 262, 259, 260 )
        (   32, 232, 237, 270, 264, 264, 259, 262, 260, 260 )
        (   34, 232, 235, 270, 264, 265, 259, 263, 259, 261 )
        (   27, 231, 238, 267, 263, 264, 259, 262, 258, 260 )
        (   34, 235, 240, 266, 263, 262, 259, 261, 258, 260 )
        (   16, 228, 237, 265, 263, 263, 260, 261, 259, 260 )
        (   15, 227, 234, 265, 265, 264, 259, 262, 259, 261 )
        (   32, 233, 272, 263, 264, 262, 260, 260, 258, 260 )
```

Class 3

```
(   28, 119, 229, 233, 267, 265, 265, 263, 261, 261 )
(   29, 118, 253, 232, 268, 265, 264, 264, 260, 263 )
(   29, 122, 231, 232, 269, 265, 263, 264, 260, 262 )
(   31, 123, 232, 233, 270, 265, 263, 264, 261, 262 )
(   35, 124, 232, 233, 271, 266, 263, 264, 262, 261 )
(   40, 225, 232, 233, 271, 265, 263, 263, 264, 258 )
(   41, 227, 231, 234, 269, 266, 264, 263, 263, 260 )
(   40, 226, 233, 277, 239, 266, 264, 260, 264, 262 )
(   21, 118, 284, 233, 269, 265, 261, 266, 260, 262 )
(   23, 122, 227, 234, 267, 265, 261, 266, 259, 262 )
(   19, 117, 226, 235, 266, 265, 261, 266, 259, 262 )
(   15, 104, 226, 236, 265, 265, 264, 265, 259, 263 )
(    4,  92, 227, 236, 266, 264, 266, 263, 261, 262 )
(    1,  86, 226, 239, 266, 264, 266, 261, 263, 260 )
(    1,  81, 227, 273, 264, 265, 267, 260, 262, 262 )
(  191,  78, 229, 271, 264, 266, 266, 260, 262, 261 )
```

2. PHASE 2 : 421 SAMPLE TEST

Read each class mean as a column vector.

	class		
component	1	2	3
a	121.479	49.766	34.596
b	133.914	188.723	185.349
c	240.907	242.022	239.630
d	266.378	263.657	245.575
e	264.964	264.482	264.459
f	263.121	263.350	264.849
g	261.736	261.562	263.959
h	260.564	260.693	261.685
i	260.586	260.058	261.384
j	259.650	260.095	261.555

B. REFERENCE POINT TEST

REFERENCE POINT	I
1. 48 SAMPLE TEST	
a. Iteration 1	
(9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999)	1.592025
(-9999, -9999, -9999, -9999, -9999, - 9999, -9999, -9999, -9999, -9999)	1.551202
(-5000, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999)	1.034770
(-1000, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999)	0.995774
(- 500, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999)	1.014196
(0, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999)	1.035598
(750, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999)	1.078592
(3000, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999)	1.269396
(7000, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999, -9999)	1.833402
b. ITERATION 2	
(-3000, -9999, -9999, -9999, -9999,	

```
                  -9999, -9999, -9999, -9999, -9999 )       0.974786
( -2000, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.975093
( -1500, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.982432
( -1299, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.990013
( - 900, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.998687
( - 700, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       1.005186
```

c. ITERATION 3

```
( -5000, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       1.034770
( -3300, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.978489
( -3500, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.981921
( -3700, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.985854
( -3900, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.991339
( -4100, -9999, -9999, -9999, -9999,
                  -9999, -9999, -9999, -9999, -9999 )       0.997118
```

```
( -4400, -9999, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )     1.007870
```

d. ITERATION 4

```
( -9999, -5000, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )     2.583654
( -9999, -1000, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )     7.689868
( -9999, - 500, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )     9.428020
( -9999,     0, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )    11.793727
( -9999,   750, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )    17.107452
( -3000, 3000, -9999, -9999, -9999,
           -9999, -9999, -9999, -9999, -9999 )     73.984070
( -9999,  7000, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )    50.770370
( -3000, -5000, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )     0.754902
( -3000, -1000, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )     1.228729
( -3000, - 500, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )     1.515115
( -3000,     0, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )     1.952134
```

```
( -3000,   750, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     3.096567
( -3000,  3000, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )    24.588181
( -3000,  7000, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )    33.357605
```

e. ITERATION 5

```
( -3000, -9500, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     0.949582
( -3000, -9000, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     0.925911
( -3000, -8500, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     0.901211
( -3000, -8000, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     0.876356
( -3000, -7500, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     0.851815
( -3000, -7000, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     0.828641
( -3000, -6500, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     0.806097
( -3000, -6000, -9999, -9999, -9999,
             -9999, -9999, -9999, -9999, -9999 )     0.785749
( -3000, -5500, -9999, -9999, -9999,
```

```
            -9999, -9999, -9999, -9999, -9999 )      0.768541
( -3000, -4500, -9999, -9999, -9999
            -9999, -9999, -9999, -9999, -9999 )      0.747745
( -3000, -4000, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      0.747534
```

f. ITERATION 6

```
( -3000, -3600, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      0.755150
( -3000, -3200, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      0.772826
( -3000, -2800, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      0.801493
( -3000, -2400, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      0.847168
( -3000, -2200, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      0.878674
( -3000, -1800, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      0.956538
( -3000, -1400, -9999, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.069383
```

g. ITERATION 7

```
( -9999, -9999, -7000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.733785
```

```
( -9999, -9999, -3000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      2.066772
( -9999, -9999, -   1, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      2.410366
( -9999, -9999,  4000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      3.048544
( -9999, -9999,  8000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      3.992909
( -3000, -9999, -7000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.071215
( -3000, -9999, -3000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.261718
( -3000, -9999, -   1, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.467827
( -3000, -9999,  4000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.874722
( -3000, -4500,  8000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      2.517179
( -3000, -4500, -7000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      0.815090
( -3000, -4500, -3000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.023657
( -3000, -4500, -   1, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.346619
( -3000, -4500,  4000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      2.233199
```

```
( -3000, -4500, 8000, -9999, -9999,
            -9999, -9999, -9999, -9999, -9999 )      4.224475
```

h. ITERATION 8

```
( -3000, -4500, -9999, -7000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.772202
( -3000, -4500, -9999, -5000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.954675
( -3000, -4500, -9999, -3000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      2.171665
( -3000, -4500, -9999, -1000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      2.430396
( -3000, -4500, -9999, -    1, -9999,
            -9999, -9999, -9999, -9999, -9999 )      2.582221
( -3000, -4500, -9999,  2000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      2.937319
( -3000, -4500, -9999,  4000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      3.374505
( -3000, -4500, -9999,  6000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      3.919746
( -3000, -4500, -9999,  8000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      4.616714
( -3000, -4500, -9999, -9000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.552190
( -3000, -4500, -9999, -7000, -9999,
```

```
            -9999, -9999, -9999, -9999, -9999 )      1.553089
( -3000, -4500, -9999, -5000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.554962
( -3000, -4500, -9999, -3000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.556623
( -3000, -4500, -9999, -1000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.559961
( -3000, -4500, -9999, -    1, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.561696
( -3000, -4500, -9999,  2000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.564777
( -3000, -4500, -9999,  4000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.569691
( -3000, -4500, -9999,  8000, -9999,
            -9999, -9999, -9999, -9999, -9999 )      1.580138

         1. ITERATION 9


( -9999, -9999, -9999, -9999, -9999,
            -9000, -9999, -9999, -9999, -9999 )      1.543766
( -9999, -9999, -9999, -9999, -9999,
            -7000, -9999, -9999, -9999, -9999 )      1.526525
( -9999, -9999, -9999, -9999, -9999,
            -5000, -9999, -9999, -9999, -9999 )      1.509349
( -9999, -9999, -9999, -9999, -9999,
            -3000, -9999, -9999, -9999, -9999 )      1.492971
```

```
( -9999, -9999, -9999, -9999, -9999,
          -1000, -9999, -9999, -9999, -9999 )      1.477160
( -9999, -9999, -9999, -9999, -9999,
          -   1, -9999, -9999, -9999, -9999 )      1.468720
( -9999, -9999, -9999, -9999, -9999,
           2000, -9999, -9999, -9999, -9999 )      1.452930
( -9999, -9999, -9999, -9999, -9999,
           4000, -9999, -9999, -9999, -9999 )      1.437694
( -9999, -9999, -9999, -9999, -9999,
           6000, -9999, -9999, -9999, -9999 )      1.422396
( -9999, -9999, -9999, -9999, -9999,
           8000, -9999, -9999, -9999, -9999 )      1.407335
( -9999, -9999, -9999, -9999, -9999,
          -9999, -9000, -9999, -9999, -9999 )      1.544096
( -9999, -9999, -9999, -9999, -9999,
          -9999, -7000, -9999, -9999, -9999 )      1.527735
( -9999, -9999, -9999, -9999, -9999,
          -9999, -5000, -9999, -9999, -9999 )      1.511520
( -9999, -9999, -9999, -9999, -9999,
          -9999, -3000, -9999, -9999, -9999 )      1.495849
( -9999, -9999, -9999, -9999, -9999,
          -9999, -1000, -9999, -9999, -9999 )      1.481759
( -9999, -9999, -9999, -9999, -9999,
          -9999, -   1, -9999, -9999, -9999 )      1.474133
( -9999, -9999, -9999, -9999, -9999,
          -9999,  2000, -9999, -9999, -9999 )      1.459345
```

```
( -9999, -9999, -9999, -9999, -9999,
            -9999,  4000, -9999, -9999, -9999 )      1.445719
( -9999, -9999, -9999, -9999, -9999,
            -9999,  6000, -9999, -9999, -9999 )      1.431514
( -9999, -9999, -9999, -9999, -9999,
            -9999,  8000, -9999, -9999, -9999 )      1.418290
( -9999, -9999, -9999, -9999, -9999,
            -9999, -9999, -9000, -9999, -9999 )      1.543633
( -9999, -9999, -9999, -9999, -9999,
            -9999, -9999, -7000, -9999, -9999 )      1.529531
( -9999, -9999, -9999, -9999, -9999,
            -9999, -9999, -5000, -9999, -9999 )      1.484761
```

j. ITERATION 10

```
( -9999, -9999, -9999, -9999, -9999,
            -9999, -9999, -    1, -9999, -9999 )      1.477756
( -9999, -9999, -9999, -9999, -9999,
            -9999, -9999,  2000, -9999, -9999 )      1.464210
( -9999, -9999, -9999, -9999, -9999,
            -9999, -9999,  4000, -9999, -9999 )      1.450755
( -9999, -9999, -9999, -9999, -9999,
            -9999, -9999,  6000, -9999, -9999 )      1.437445
( -9999, -9999, -9999, -9999, -9999,
            -9999, -9999,  8000, -9999, -9999 )      1.424325
( -9999, -9999, -9999, -9999, -9999,
```

```
              -9999, -9999, -9999, -9000, -9999 )       1.545251
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999, -7000, -9999 )       1.532997
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999, -5000, -9999 )       1.520181
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999, -3000, -9999 )       1.507595
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999, -1000, -9999 )       1.495237
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999, -   1, -9999 )       1.489653
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999,  2000, -9999 )       1.478065
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999,  4000, -9999 )       1.466319
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999,  6000, -9999 )       1.455643
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999,  8000, -9999 )       1.444107
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999, -9999, -9000 )       1.547930
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999, -9999, -7000 )       1.538848
( -9999, -9999, -9999, -9999, -9999,
              -9999, -9999, -9999, -9999, -5000 )       1.531022
( -9999, -9999, -9999, -9999, -9999,
```

```
          -9999, -9999, -9999, -9999, -3000 )     1.523348
( -9999, -9999, -9999, -9999, -9999,
          -9999, -9999, -9999, -9999, -1000 )     1.514919
( -9999, -9999, -9999, -9999, -9999,
          -9999, -9999, -9999, -9999, -   1 )     1.510691
( -9999, -9999, -9999, -9999, -9999,
          -9999, -9999, -9999, -9999,  2000 )     1.502452
( -9999, -9999, -9999, -9999, -9999, -9999,
          -9999, -9999, -9999, -9999,  4000 )     1.494842
( -9999, -9999, -9999, -9999, -9999, -9999,
          -9999, -9999, -9999, -9999,  6000 )     1.486907
( -9999, -9999, -9999, -9999, -9999, -9999,
          -9999, -9999, -9999, -9999,  8000 )     1.480084
```

k. ITERATION 11

```
( -3000, -4500, -9999, -9999, -9999,
           9999,  9999,  9999,  9999,  9999 )     0.442269
( -3000, -4500, -9999, -9999, -9999,
          -9999, -9999, -9999, -9999, -9999 )     0.747745
( -3000, -4500, -7000, -9999, -9999,
          -9999, -9999, -9999, -9999, -9999 )     0.815090
( -3000, -4500, -7500, -9999, -9999,
          -9999, -9999, -9999, -9999, -9999 )     0.800422
( -3000, -4500, -8000, -9999, -9999,
          -9999, -9999, -9999, -9999, -9999 )     0.786349
```

```
( -3000, -4500, -7000, -9999, -9999,
            9999,  9999,  9999,  9999,  9999 )     0.452415
( -3000, -4500, -7000, -9999, -9999,
           99999, 99999, 99999, 99999, 99999 )     0.189089
```

2. 421 SAMPLE TEST

a. ITERATION 1

```
( -3000, -4500, -7000, -9999, -9999,
           99999, 99999, 99999, 99999, 99999 )     1.578162
( -3000. -4500, -7000,-99999,-99999,
           99999, 99999, 99999, 99999, 99999 )     1.310979
```

APPENDIX G TEST DATA FROM CASE STUDY 7

A. SAMPLE DATA

The class means only are provided. Read each mean as a column vector.

vector component	class		
	1	2	3
a	121.479	49.766	34.596
b	133.914	188.723	185.349
c	240.907	242.022	239.630
d	266.378	263.657	245.575
e	264.964	264.482	264.459
f	263.121	263.350	264.849
g	261.736	261.562	263.959
h	260.564	260.693	261.685
i	260.586	260.058	261.384
j	259.650	260.095	261.555
k	259.635	259.328	260.370
l	258.836	259.073	259.870
m	258.828	258.832	259.609
n	258.621	258.606	259.301
o	258.436	258.423	259.185
p	258.378	258.233	259.014
q	258.100	258.306	259.021
r	258.100	257.963	258.507
s	257.878	258.044	258.322
t	257.836	257.613	258.178
u	257.764	257.788	258.164
v	257.678	257.576	258.109
w	257.607	257.628	258.021
x	257.421	257.343	257.801
y	257.414	257.365	257.829
z	257.343	257.292	257.788
*	257.135	257.255	257.774
@	257.200	257.219	257.651
#	257.128	257.095	257.637
$	257.107	257.102	257.479
%	257.078	257.051	257.418
&	257.050	257.095	257.493

B. REFERENCE POINT DATA

Read each reference point as a column vector. The value of a vector component across a row remain constant unless indicated otherwise. The last component value change to occur remains in effect until modified. The resulting (I) is shown below the corresponding vector.

1. ITERATION 1

a	-3000	...	...	...	...
b	-4500	...	...	...	...
c	-7000	...	...	...	...
d	-9999	...	...	...	...
e	-9999	...	...	...	...
f	99999	...	...	...	...
g	99999	...	...	...	...
h	99999	...	...	...	...
i	9999	99999	...	9999	...
j	9999	99999	...	9999	...
k	9999	-9999	99999	9999	99999
l	9999	-9999	99999	9999	99999
m	9999	...	...	99999	-9999
n	9999	...	...	...	-99999
o	9999	...	...	...	-99999
p	9999	...	...	...	-99999
q	9999	...	...	99999	99999
r	9999	...	...	99999	99999
s	9999	...	...	-9999	99999
t	9999	...	...	-9999	99999
u	9999	...	...	-9999	99999
v	9999	...	...	-9999	-99999
w	9999	...	...	-9999	-99999
x	9999	...	...	-9999	-99999
y	9999	...	...	-9999	99999
z	9999	...	...	-9999	99999
*	9999	...	...	-9999	99999
@	9999	...	...	-9999	99999
#	9999	...	...	-9999	99999
$	9999	...	...	-9999	-99999
%	9999	...	...	-9999	-99999
&	9999	...	...	-9999	-99999
	1.683570	1.844750	3.736396	1.896465	54.177185

2. ITERATION 2

```
a      -3000      ...       ...       ...
b      -4500      ...       ...       ...
c      -7000      ...       ...       ...
d      -9999      ...       ...       ...
e      -9999      ...       ...       ...
f      99999      ...       ...       ...
g      99999      ...       ...       ...
h      99999      ...       ...       ...
i      99999      ...       ...       ...
j      99999      ...       ...       ...
k      -6000     -2000      3000      7000
l      -9999      ...       ...       ...
m      -9999      ...       ...       ...
n      -9999      ...       ...       ...
o      -9999      ...       ...       ...
p      -9999      ...       ...       ...
q      -9999      ...       ...       ...
r      -9999      ...       ...       ...
s      -9999      ...       ...       ...
t      -9999      ...       ...       ...
u      -9999      ...       ...       ...
v      -9999      ...       ...       ...
w      -9999      ...       ...       ...
x      -9999      ...       ...       ...
y      -9999      ...       ...       ...
z      -9999      ...       ...       ...
*      -9999      ...       ...       ...
@      -9999      ...       ...       ...
#      -9999      ...       ...       ...
$      -9999      ...       ...       ...
%      -9999      ...       ...       ...
&      -9999      ...       ...       ...

   1.828800  1.748404  1.877700  1.802315
```

3. ITERATION 3

a	-3000	...	...	...	...
b	-4500	...	...	...	...
c	-7000	...	...	...	...
d	-9999	...	...	...	...
e	-9999	...	...	...	...
f	99999	...	...	...	...
g	99999	...	...	...	...
h	99999	...	...	...	...
i	99999	...	...	...	...
j	99999	...	...	...	...
k	-9999	...	...	...	...
l	-6000	-2000	3000	7000	-9999
m	-9999	...	...	...	-6000
n	-9999	...	...	...	...
o	-9999	...	...	...	...
p	-9999	...	...	...	...
q	-9999	...	...	...	...
r	-9999	...	...	...	...
s	-9999	...	...	...	...
t	-9999	...	...	...	...
u	-9999	...	...	...	...
v	-9999	...	...	...	...
w	-9999	...	...	...	...
x	-9999	...	...	...	...
y	-9999	...	...	...	...
z	-9999	...	...	...	...
*	-9999	...	...	...	...
@	-9999	...	...	...	...
#	-9999	...	...	...	...
$	-9999	...	...	...	...
%	-9999	...	...	...	...
&	-9999	...	...	...	...
	1.823965	1.772686	1.854467	1.785567	1.821434

4. ITERATION 4

```
a     -3000     ...       ...       ...       ...
b     -4500     ...       ...       ...       ...
c     -7000     ...       ...       ...       ...
d     -9999     ...       ...       ...       ...
e     -9999     ...       ...       ...       ...
f     99999     ...       ...       ...       ...
g     99999     ...       ...       ...       ...
h     99999     ...       ...       ...       ...
i     99999     ...       ...       ...       ...
j     99999     ...       ...       ...       ...
k     -9999     ...       ...       ...       ...
l     -9999     ...       ...       ...       ...
m     -2000      3000      7000     -9999     ...
n     -9999     ...       ...       -6000     -2000
o     -9999     ...       ...       ...       ...
p     -9999     ...       ...       ...       ...
q     -9999     ...       ...       ...       ...
r     -9999     ...       ...       ...       ...
s     -9999     ...       ...       ...       ...
t     -9999     ...       ...       ...       ...
u     -9999     ...       ...       ...       ...
v     -9999     ...       ...       ...       ...
w     -9999     ...       ...       ...       ...
x     -9999     ...       ...       ...       ...
y     -9999     ...       ...       ...       ...
z     -9999     ...       ...       ...       ...
*     -9999     ...       ...       ...       ...
@     -9999     ...       ...       ...       ...
#     -9999     ...       ...       ...       ...
$     -9999     ...       ...       ...       ...
%     -9999     ...       ...       ...       ...
&     -9999     ...       ...       ...       ...

   1.763098  1.882494  1.795417  1.822107  1.754014
```

5. ITERATION 5

```
a     -3000      ...        ...        ...        ...
b     -4500      ...        ...        ...        ...
c     -7000      ...        ...        ...        ...
d     -9999      ...        ...        ...        ...
e     -9999      ...        ...        ...        ...
f     99999      ...        ...        ...        ...
g     99999      ...        ...        ...        ...
h     99999      ...        ...        ...        ...
i     99999      ...        ...        ...        ...
j     99999      ...        ...        ...        ...
k     -9999      ...        ...        ...        ...
l     -9999      ...        ...        ...        ...
m     -9999      ...        ...        ...        ...
n      3000      7000       -9999      ...        ...
o     -9999      -9999      -6000      -2000       3000
p     -9999      ...        ...        ...        ...
q     -9999      ...        ...        ...        ...
r     -9999      ...        ...        ...        ...
s     -9999      ...        ...        ...        ...
t     -9999      ...        ...        ...        ...
u     -9999      ...        ...        ...        ...
v     -9999      ...        ...        ...        ...
w     -9999      ...        ...        ...        ...
x     -9999      ...        ...        ...        ...
y     -9999      ...        ...        ...        ...
z     -9999      ...        ...        ...        ...
*     -9999      ...        ...        ...        ...
@     -9999      ...        ...        ...        ...
#     -9999      ...        ...        ...        ...
$     -9999      ...        ...        ...        ...
%     -9999      ...        ...        ...        ...
&     -9999      ...        ...        ...        ...

    1.816326   1.762993   1.802553   1.762321    1.868320
```

6. ITERATION 6

```
a      -3000      ...        ...        ...        ...
b      -4500      ...        ...        ...        ...
c      -7000      ...        ...        ...        ...
d      -9999      ...        ...        ...        ...
e      -9999      ...        ...        ...        ...
f      99999      ...        ...        ...        ...
g      99999      ...        ...        ...        ...
h      99999      ...        ...        ...        ...
i      99999      ...        ...        ...        ...
j      99999      ...        ...        ...        ...
k      -9999      ...        ...        ...        ...
l      -9999      ...        ...        ...        ...
m      -9999      ...        ...        ...        ...
n      -9999      ...        ...        ...        ...
o       7000     -9999       ...        ...        ...
p      -9999     -6000      -2000       3000       7000
q      -9999      ...        ...        ...        ...
r      -9999      ...        ...        ...        ...
s      -9999      ...        ...        ...        ...
t      -9999      ...        ...        ...        ...
u      -9999      ...        ...        ...        ...
v      -9999      ...        ...        ...        ...
w      -9999      ...        ...        ...        ...
x      -9999      ...        ...        ...        ...
y      -9999      ...        ...        ...        ...
z      -9999      ...        ...        ...        ...
*      -9999      ...        ...        ...        ...
@      -9999      ...        ...        ...        ...
#      -9999      ...        ...        ...        ...
$      -9999      ...        ...        ...        ...
%      -9999      ...        ...        ...        ...
&      -9999      ...        ...        ...        ...


   1.793631  1.789902  1.758676   1.803580  1.771052
```

7. ITERATION 7

```
a      -3000      ...        ...        ...
b      -4500      ...        ...        ...
c      -7000      ...        ...        ...
d      -9999      ...        ...        ...
e      -9999      ...        ...        ...
f      99999      ...        ...        ...
g      99999      ...        ...        ...
h      99999      ...        ...        ...
i      99999      ...        ...        ...
j      99999      ...        ...        ...
k      -9999      ...        ...        ...
l      -9999      ...        ...        ...
m      -9999      ...        ...        ...
n      -9999      ...        ...        ...
o      -9999      ...        ...        ...
p      -9999      ...        ...        ...
q      -6000     -2000       3000       7000
r      -9999      ...        ...        ...
s      -9999      ...        ...        ...
t      -9999      ...        ...        ...
u      -9999      ...        ...        ...
v      -9999      ...        ...        ...
w      -9999      ...        ...        ...
x      -9999      ...        ...        ...
y      -9999      ...        ...        ...
z      -9999      ...        ...        ...
*      -9999      ...        ...        ...
@      -9999      ...        ...        ...
#      -9999      ...        ...        ...
$      -9999      ...        ...        ...
%      -9999      ...        ...        ...
&      -9999      ...        ...        ...

    1.787084  1.723260  1.808782  1.764592
```

8. ITERATION 8

```
a       -3000        ...         ...         ...
b       -4500        ...         ...         ...
c       -7000        ...         ...         ...
d       -9999        ...         ...         ...
e       -9999        ...         ...         ...
f       99999        ...         ...         ...
g       99999        ...         ...         ...
h       99999        ...         ...         ...
i       99999        ...         ...         ...
j       99999        ...         ...         ...
k       -9999        ...         ...         ...
l       -9999        ...         ...         ...
m       -9999        ...         ...         ...
n       -9999        ...         ...         ...
o       -9999        ...         ...         ...
p       -9999        ...         ...         ...
q       -9999        ...         ...         ...
r       -2000       -1500       -1000         -500
s       -9999        ...         ...         ...
t       -9999        ...         ...         ...
u       -9999        ...         ...         ...
v       -9999        ...         ...         ...
w       -9999        ...         ...         ...
x       -9999        ...         ...         ...
y       -9999        ...         ...         ...
z       -9999        ...         ...         ...
*       -9999        ...         ...         ...
@       -9999        ...         ...         ...
#       -9999        ...         ...         ...
$       -9999        ...         ...         ...
%       -9999        ...         ...         ...
&       -9999        ...         ...         ...

   1.734462   1.764483   1.706574   1.782977
```

9. ITERATION 9

```
a      -3000      ...        ...        ...
b      -4500      ...        ...        ...
c      -7000      ...        ...        ...
d      -9999      ...        ...        ...
e      -9999      ...        ...        ...
f      99999      ...        ...        ...
g      99999      ...        ...        ...
h      99999      ...        ...        ...
i      99999      ...        ...        ...
j      99999      ...        ...        ...
k      -9999      ...        ...        ...
l      -9999      ...        ...        ...
m      -9999      ...        ...        ...
n      -9999      ...        ...        ...
o      -9999      ...        ...        ...
p      -9999      ...        ...        ...
q      -9999      ...        ...        ...
r      -9999      ...        ...        ...
s      -3000      -2000      -1000      -    1
t      -9999      ...        ...        ...
u      -9999      ...        ...        ...
v      -9999      ...        ...        ...
w      -9999      ...        ...        ...
x      -9999      ...        ...        ...
y      -9999      ...        ...        ...
z      -9999      ...        ...        ...
*      -9999      ...        ...        ...
@      -9999      ...        ...        ...
#      -9999      ...        ...        ...
$      -9999      ...        ...        ...
%      -9999      ...        ...        ...
&      -9999      ...        ...        ...

   1.895550   1.733106   1.705211   1.811335
```

10. ITERATION 10

```
a      -3000       ...
b      -4500       ...
c      -7000       ...
d      -9999       ...
e      -9999       ...
f      99999       ...
g      99999       ...
h      99999       ...
i      99999       ...
j      99999       ...
k      -1000      -2000
l      -1000      -2000
m      -1000      -2000
n      -1000      -2000
o      -1000      -2000
p      -1000      -2000
q      -1000      -2000
r      -1000      -2000
s      -1000      -2000
t      -1000      -2000
u      -1000      -2000
v      -1000      -2000
w      -1000      -2000
x      -1000      -2000
y      -1000      -2000
z      -1000      -2000
*      -1000      -2000
@      -1000      -2000
#      -1000      -2000
$      -1000      -2000
%      -1000      -2000
&      -1000      -2000

    1.644567    1.624708
```

```
C     MAIN PROGRAM - SEARCHR2
C     THIS MODULE CONTROLS PROGRAM FLOW.
C     WITHIN THIS MODULE, THE DATA POINTS ARE GENERATED,  FROM
C     WHICH THE DIFFERENCE MATRIX  WILL BE DETERMINED.
C
C*****************************************************************
C     THE MODULE IS NOW CONFIGURED FOR EXHAUSTIVE 3 SPACE IN THE
C     RANGE OF 0 - 7
C*****************************************************************
C
C     INPUT - AS PRESENTLY WRITTEN,  THERE IS NO INPUT AS ALL DATA
C             POINTS ARE GENERATED WITHIN THIS MODULE.
C     OUT PUT - THIS MODULE PRESENTLY HAS NO OUTPUT
C
C     VARIABLES DEFINITION
C
C     SAMPLE - AN ARRAY OF DATA POINTS.  DATA IS EITHER GENERATED
C              INTERNALLY TO THIS MODULE OR PROVIDED BY THE USER
C     NUMSAM - NUMBER OF SAMPLES
C     NUMFEA - NUMBER OF FEATURES PER SAMPLE, IE, THE DIMENSIONALITY
C     REF2 -   VECTOR CONTAINING THE SECOND REFERENCE POINT.
C              INITIALIZED TO 0, ITS VALUE WILL BE FIXED BY REFTWO S/R
C     LENG  -  THE NUMBER OF DIFFERENCE VECTORS CONTAINED IN THE
C              DIFFERENCE MATRIX. ITS VALUE IS DETERMINED IN  STATEMENT
```

```
C              155.                                                    SEA00250
C        DUMMY  - A UTILITY ARRAY FOR GENERAL USE.  IN S/R REFTWO IT WILL SEA00260
C                 CONTAIN THE DIFFERENCE MATRIX                        SEA00270
C        UTILE1 - UTILITI ARRAY FOR GENERAL USE.  IN S/R REFTWO IT WILL  SEA00271
C                 CONTAIN THE DIFFERENCES IN DISTANCE SQUARED MATRIX   SEA00272
C                                                                      SEA00280
C        STEPS IN PROCESSING ARE                                       SEA00290
C                                                                      SEA00300
C         1.  GENERATE DATA POINTS                                     SEA00310
C         2. SET PARAMETERS FOR 1ST SUBROUTINE CALL                    SEA00320
C         3. CALL REFTWO SUBROUTINE TO DETERMINE  ALL POSSIBLE  REF 2  SEA00330
C            POINTS IN THE SPECIFIED SEARCH RANGE                      SEA00340
C                                                                      SEA00350
C        PROCEDURE TO MODIFY CODE FOR VARIOUS SAMPLE SIZES             SEA00360
C                                                                      SEA00370
C        0.  FOR DATA SPACES WITH RANGES OF GREATER THAN  127 CHANGE ALL SEA00380
C            INTEGER*2 DECLARATIONS TO INTEGER*4                       SEA00390
C        1.  CHANGE DIMENSION STATEMENT TO REFLECT DESIRED NUMBER OF   SEA00400
C            SAMPLES,FEATURES, AND REF2                                SEA00410
C        2.  ADD DO LOOPS AND SAMPLES ELEMNTS TO ACHIEVE DESIRED       SEA00420
C            DIMENSIONALITY AFTER STATEMENT 100                        SEA00430
C        3.  MODIFY COMMENT CARDS TO REFLECT NEW STATUS.               SEA00440
C        4.  VERIFY ALL SUBROUTINES ARE MODIFIED TO PROCESS NEW DATA SPACE SEA00450
C        5.  VERIFY THE DUMMY MATRIX IS AS LARGE AS THE  LENG AND NUMBER  SEA00460
```

```
C           OF FEATURES IN THE DIFFERENCE MATRIX IN S/R REFTWO                  SEA00470
C       6.  MODIFY STATEMENT 90 TO DESIRED NUMBER OF SAMPLES                    SEA00480
C       7.  MODIFY STATEMENT 91 TO DESIRED NUMBER OF FEATURES                   SEA00490
C                                                                               SEA00500
C       CONTROL MODULE                                                          SEA00510
C                                                                               SEA00520
        INTEGER*2 SAMPLE,REF2    ,I,J,K,L,DUMMY                                 SEA00530
        INTEGER*4 LENG,NUMSAM,NUMFEA                                            SEA00540
        DIMENSION SAMPLE(1000,3),DUMMY(499500,3),REF2(3)                        SEA00550
C                                                                               SEA00560
C       INITIALIZE DATA                                                         SEA00570
C                                                                               SEA00580
   90   NUMSAM = 343                                                            SEA00590
   91   NUMFEA = 3                                                              SEA00600
C                                                                               SEA00610
C       GENERATE DATA POINTS                                                    SEA00620
C                                                                               SEA00630
  100   L = 1                                                                   SEA00640
        DO 150  I = 1,8                                                         SEA00650
          DO 150  J = 1,8                                                       SEA00660
            DO 150  K = 1,8                                                     SEA00670
              SAMPLE (L,1) = I - 1                                              SEA00680
              SAMPLE (L,2) = J - 1                                              SEA00690
              SAMPLE (L,3) = K - 1                                              SEA00700
```

```
  150 L = L + 1                                                      SEA00710
C                                                                    SEA00720
C     SET PARAMETERS THEN DETERMINE REFERENCE POINT 2                SEA00730
C                                                                    SEA00740
  155 LENG = (NUMSAM * (NUMSAM - 1)) / 2                             SEA00750
      DO 160  I = 1, NUMFEA                                          SEA00760
  160   REF2(I) = 0                                                  SEA00770
      CALL  REFTWO (SAMPLE,NUMSAM,NUMFEA,LENG,REF2,DUMMY)            SEA00780
      STOP                                                           SEA00790
      END                                                            SEA00800
C     S/R REFTWO DETERMINES THE DIFFERENCE MATRIX AND THEN  ITERATIVELY SEA00810
C     SEARCHES FOR  VALID VALUES OF REFERENCE POINT 2. WHEN A VALID  SEA00820
C     POINT IS FOUND, IT IS IMMEDIATELY OUTPUT TO THE PRINTER.       SEA00830
C                                                                    SEA00840
C     INPUT -  NUMBER OF SAMPLES                                     SEA00850
C              NUMBER OF FEATURES PER SAMPLE                         SEA00860
C              LENGTH OF THE DIFFERENCE MATRIX                       SEA00870
C              DUMMY VALUE FOR REFERENCE POINT 2                     SEA00880
C              DUMMY ARRAY TO USE FOR DIFFERENCE MATRIX              SEA00890
C                                                                    SEA00900
C     OUTPUT   NUMBER OF SAMPLES, UNMODIFIED FROM INPUT              SEA00910
C              NUMBER OF FEATURES, UNMODIFIED FROM INPUT             SEA00920
C              LENGTH OF  DIFF. MATRIX, UNMODIFIED FROM INPUT        SEA00930
C              REF2 CONTAINS THE LAST VALID VALUE TESTED DURING THE  SEA00940
```

```
C               ITERATIVE SEARCH.  ITS VALUE MAY OR MAY NOT BE A VALID    SEA00950
C               REFERENCE POINT 2                                         SEA00960
C               DUMMY ARRAY CONTAINING DIFFERENCES                        SEA00970
C                                                                         SEA00980
C         VARIABLES DEFINED                                               SEA00990
C                                                                         SEA01000
C         SAMPLE    ARRAY OF SAMPLE DATA POINTS                           SEA01010
C         NUMSAM    NUMBER OF SAMPLE VECTORS PASSED IN                    SEA01020
C         NUMFEA    NUMBER OF FEATURES PER SAMPLE                         SEA01030
C         LENG      NUMBER OF ELEMENTS IN DIFFERENCE MATRIX (SEE DIFER)   SEA01040
C         REF2      REFERENCE POINT 2 VECTOR, INPUT VALUE = DON'T CARE,   SEA01050
C                   FIRST USED TO CONTAIN A DATA POINT VALUE FROM WHICH   SEA01060
C                   DIFFERENCES TO ALL OTHER POINTS ARE COMPUTED.         SEA01070
C                   BEYOND STATEMENT 320 SECOND USE IS AS A CANDIDATE     SEA01071
C                   REFERENCE POINT.                                      SEA01072
C                   OUTPUT VALUE IS LAST TESTED  POINT. IT MAY OR MAY NOT BE SEA01080
C                   A VALID REF 2 POINT.                                  SEA01090
C         DIFER     ARRAY OF DIFFERENCES BETWEEN ALL POINTS IN THE DATA SPACESEA01100
C                   NOTE DIFER CONTAINS ONLY THE LOWER TRIANGULAR PORTION OF SEA01110
C                   THE MATRIX, THE UPPER TRIANGULAR PORTION IS THE NEGATIVE SEA01120
C                   OF THE LOWER, AND THE MAIN DIAGONAL CONTAINS ZEROS.   SEA01130
C         KEY1      CONTROL CONSTANT = 1, USED AS BEGINNING INDEX FOR ALL DO SEA01140
C                   LOOPS                                                 SEA01150
C         INDEX1    COUNTER OF THE NUMBER OF SAMPLES FOR WHICH DIFFERENCES  SEA01160
```

```
C               HAVE BEEN COMPUTED
C         INDX      POINTER TO THE NEXT BUFFER LOCATION TO BE FILLED
C         INDX1,INDX2,KEY2  CONTROL VARIABLES FOR DO LOOPS
C         TEMP1     SUMMATION OF A MULTIPLICATION OF A DIFFERENCE VECTOR
C                   AND REF PT 2
C         TEMP2     SUMMATION OF MULTIPLICATION OF A NEGATIVE DIFFERENCE
C                   VECTOR AND REF PT 2
C         FLAG      A LOGICAL VALUE USED TO QUE WRITE STATEMENT 355
C                   TRUE IMPLIES NO VALID REF PT 2 WAS FOUND WITHIN THE
C                   TEST RANGE
C                   FALSE IMPLIES AT LEAST ONE VALID REF PT 2 HAS BEEN FOUND
C                   WITHIN THE TEST RANGE
C         SWITCH     EQUALS ONE PROVIDES AN OUTPUT OF THE DIFFERENCE MATRIX
C                    NOT EQUAL ONE SUPPRESSES THE OUTPUT OF THE DIFFERENCE
C                    MATRIX
C                    SWITCH IS SET IN THE INTEGER*2 DECLARATION STATEMENT
C                    OF THIS SUBROUTINE
C
C         STEPS IN PROCESSING ARE
C
C         1.  COMPUTE DIFFERENCE MATRIX
C         2.  SEARCH FOR VALID REFERENCE POINTS 2 BY ITERATIVE METHOD
C
C         PROCEDURE TO MODIFY CODE FOR VARIOUS SAMPLE SIZES
```

```
C                                                                       SEA01410
C     1.  MODIFY FORMAT STATEMENT NUMBER 340 TO REFLECT CORRECT  NUMFEA SEA01420
C     2.  MODIFY STATEMENT 300 BY ADDING THE DESIRED NUMBER OF REF2     SEA01430
C         FEATURES                                                      SEA01440
C     3.  SET THE THE DESIRED RANGE OF VALUES TO BE ITERATIVELY         SEA01450
C         SEARCHED FOR VALID REF PT 2 AT STATEMENT LABEL 300            SEA01460
C                                                                       SEA01470
C                                                                       SEA01480
      SUBROUTINE  REFTWO (SAMPLE,NUMSAM,NUMFEA,LENG,REF2,DIFER)         SEA01490
C                                                                       SEA01500
      INTEGER*4 INDX/0/,TEMP1,TEMP2,INDX1,INDX2,NUMSAM,NUMFEA           SEA01510
      INTEGER*2 SAMPLE,REF2,DIFER,REF2SQ,SAMSQ,D1DIF                    SEA01530
      INTEGER*2 KEY1/1/,INDEX/1/,II,JJ,KK,KEY2,SWITCH/0/                SEA01540
      LOGICAL FLAG/.FALSE./                                             SEA01550
      DIMENSION SAMPLE(NUMSAM,NUMFEA),DIFER(LENG,NUMFEA),REF2(NUMFEA) , SEA01520
     1 D1DIF(LENG)                                                      SEA01521
C                                                                       SEA01560
      REF2SQ = 0                                                        SEA01561
  100 DO 105  KEY2 = KEY1,NUMFEA                                        SEA01570
        REF2  (KEY2) = SAMPLE(INDEX,KEY2)                               SEA01580
 0105   REF2SQ = REF2SQ + SAMPLE(INDEX,KEY2) ** 2                       SEA01581
      INDEX = INDEX + 1                                                 SEA01590
C                                                                       SEA01600
C     COMPUTE DIFFERENCES FROM REFERENCE SAMPLE POINT TO ALL SAMPLES    SEA01610
```

```
C     WITH A HIGHER ORDER SAMPLE INDEX NUMBER                            SEA01620
C                                                                        SEA01630
      DO 205  I = INDEX,NUMSAM                                           SEA01640
        INDX = INDX + 1                                                  SEA01650
        SAMSQ = 0                                                        SEA01651
        DO 200  J = KEY1,NUMFEA                                          SEA01660
          DIFFR(INDX,J) = REF2(J) - SAMPLE(I,J)                          SEA01670
          SAMSQ = SAMSQ + SAMPLE(I,J) ** 2                               SEA01671
  200 CONTINUE                                                           SEA01680
      DIDIF(INDX) = SAMSQ - REF2SQ                                       SEA01681
      IF (SWITCH.NE.1) GO TO 205                                         SEA01690
       WRITE(6,203) (DIFFR(INDX,J),J=1,NUMFEA)                           SEA01700
  203  FORMAT(1H ,3I8)                                                   SEA01710
  205 CONTINUE                                                           SEA01720
      IF (INDEX.LE.NUMSAM - 1) GO TO 100                                 SEA01730
C                                                                        SEA01740
C     DETERMINE REF PT 2.  A POINT WILL BE A VALID TRANSFORMATION IF     SEA01750
C     REF PT 2 TIMES ALL POINTS IN THE DIFFERENCE MATRIX DOES NOT        SEA01760
C     EQUAL ZERO.                                                        SEA01770
C                                                                        SEA01780
            WRITE(6,320)                                                 SEA01790
  320 FORMAT(1H1,'THE FOLLOWING POINTS ARE VALID FOR REFERENCE  PT 2')   SEA01800
  300 DO 350  II = 1,1                                                   SEA01810
        DO 350  JJ = 1,50                                                SEA01820
```

```
         DO 350  KK = 150,250                                           SEA01830
           REF2(1) = -II                                                SEA01840
           REF2(2) = JJ                                                 SEA01850
           REF2(3) = KK                                                 SEA01860
C                                                                       SEA01870
C     SCAN THE DIFFERENCE MATRIX WITH CURRENT VALUE OF REF2             SEA01880
C                                                                       SEA01890
           DO 310  INDX1 = KEY1,LENG                                    SEA01900
             TEMP1 = 0                                                  SEA01910
             TEMP2 = 0                                                  SEA01920
             DO 305  INDX2 = KEY1,NUMFEA                                SEA01930
               TEMP1 = TEMP1 + REF2(INDX2) * DIFER(INDX1,INDX2)         SEA01940
               TEMP2 = TEMP2 + ((-DIFER(INDX1,INDX2)) * REF2(INDX2))    SEA01950
             TEMP1 = D1DIF(INDX1) + TEMP1                               SEA01951
             TEMP2 = -D1DIF(INDX1) - TEMP2                              SEA01952
  305        CONTINUE                                                   SEA01960
           IF ((TEMP1.EQ.0).OR.(TEMP2.EQ.0)) GO TO 350                  SEA01970
  310      CONTINUE                                                     SEA01980
           WRITE(6,340) (REF2(K),K = KEY1,NUMFEA)                       SEA01990
  340      FORMAT(1H ,3I8)                                              SEA02000
           FLAG = .TRUE.                                                SEA02010
  350 CONTINUE                                                          SEA02020
      IF (FLAG) GO TO 400                                               SEA02030
  355   WRITE(6,360)                                                    SEA02040
```

```
360   FORMAT(1H0,'NO VALID REF PT 2 FOUND')                                  SEA02050
400 CONTINUE                                                                 SEA02060
    RETURN                                                                   SEA02070
    END                                                                      SEA02080
```

```
C     THIS PROGRAM INPUTS DATA SAMPLES REPRESENTED IN N DIMENSIONS,
C     COMPUTES THEIR INFORMATION CONTEXT RATIO IN N SPACE, CONVERTS THE
C     SAMPLES TO M DIMENSION USING SOME REFERENCE POINTS : WHERE BOTH
C     THE M SPACE AND REFERENCE POINTS ARE USER DEFINED : AND COMPUTES
C     THEIR INFORMATION CONTEXT RATIO IN M SPACE.  THERE ARE THREE
C     MODULES IN THE PROGRAM: MAIN, SCATTR,AND REDUCE.  EACH IS SELF
C     DOCUMENTING.
C
C
C
C     THE MAIN MODULE CONTROLS DATA INPUT AND PROGRAM FLOW.  ALL
C     PARAMETERS EXCEPT FORMAT STATEMENTS FOR ALL MODULES ARE SET WITHIN
C     THE INITIALIZATION SECTION OF THIS MODULE.  THE DATA SAMPLE INPUT
C     SECTION WILL BE MODIFIED DEPENDENT ON THE INPUT MEDIUM, NUMBER
C     OF SAMPLES, AND FORMAT OF THE SAMPLES.
C      INPUTS    : SAMPLE DATA
C                  NUMBER OF REFERENCE POINTS TO PROCESS
C                  REFERENCE POINTS
C      OUTPUTS   : ALL OUT PUT PERFORMED IN THE SUBROUTINES
C
C      VARIABLES:
C               SAMPLE   ARRAY OF DATA POINTS
C               NUMSAM   NUMBER OF SAMPLES
C               NUMFEA   NUMBER OF FEATURES, IE, THE NUMBER OF
```

```
C                 DIMENSIONS
C        SAMSIZ   NUMFEA + 1, THE NUMBER OF FEATURES PLUS 1
C                 ELEMENT CONTAINING THE CLASS ID NUMBER
C        NOCLAS   NUMBER OF CLASSES WITHIN THE SAMPLE SET,
C                 CLASSES MUST BE NUMBERED CONSECUTIVELY FROM 1
C                 TO NOCLAS. ZERO IMPLIES NO CLASS ASSIGNED
C         REF2    VECTOR CONTAINING THE COORDINATES OF THE SECOND
C                 REFERENCE POINT AS DETERMINED BY PROGRAM
C                 SEARCHR2
C        NOPRNT   BOOLEAN VALUE.  TRUE IMPLIES NO PRINTING OF
C                 INPUT SAMPLES OR M SPACE REPRESENTATIONS.
C                 FALSE IMPLIES PRINT  THE INPUT DATA AND M SPACE
C                 REPRESENTATIONS
C        NOPLOT   TRUE IMPLIES NO PLOT OF M SPACE  DATA VECTORS
C                 FALSE IMPLIES OUTPUT M SPACE DATA VECTOR TO
C                 VERSATEC PLOTTER.  THIS BOOLEAN VALUE IS USED
C                 IN S/R REDUCE.
C        SAMCNT   SAMPLE COUNT:  COUNT OF NUMBER OF VALID SAMPLES
C                 READ FROM A TAPE WHICH CONTAINS BOTH VALID AND
C                 INVALID SAMPLES.  MAY NOT BE USED WITH OTHER
C                 INPUT MEDIUMS.
C        CLASS    CLASS IDENTIFICATION LABEL.  TAGS AN ELEMENT
C                 IN THE SAMPLE ARRAY.  USED IN READING DATA FROM
C                 TAPE.  THE TAG BECOMES THE LAST ELEMENT OF THE
```

```
C                         APPROPRIATE SAMPLE IN THE SAMPLE ARRAY.
C              ID         IDENTIFIES TYPE OF SAMPLE.
C                         1 IMPLIES TRAINING SAMPLE
C                         2 IMPLIES TESTING SAMPLE
C                         AN ELEMENT OF THIS ARRAY TAGS A CORRESPONDING
C                         ELEMENT IN THE SAMPLE ARRAY.  USED IN SUPPORT
C                         OF TAPE DATA FROM NWC
C              NUMREF     THE NUMBER OF REFERENCE POINTS TO BE TESTED.
C              NSPACE     THE NUMBER OF FEATURES IN THE REDUCED
C                         REPRESENTATION.  NORMALLY EQUALS 2
C              NSIZ       NSPACE + 1, THE NUMBER OF FEATURES + 1 ELEMENT
C                         CONTAINING CLASS ID NUMBER.  USED IN ARRAY
C                         ALLOCATION AND INDEXING.
C              LGCLSZ     LARGEST CLASS SIZE.  VALUE OF THE INDIVIDUAL
C                         LARGEST CLASS AS INPUT BY THE USER.  USED IN
C                         ARRAY ALLOCATION AND INDEXING.
C              BUFFER     INPUT BUFFER.  USED TO SCREEN FOR VALID /
C                         INVALID SAMPLES FROM NWC TAPE INPUT.
C              DIST1      DISTANCE 1.  IN S/R REDUCE, CONTAINS THE  X AXIS
C                         COMPONENTS OF THE 2 SPACE SAMPLES.  DECLARED
C                         HERE FOR RUN TIME ALLOCATION PURPOSES.
C              DIST2      DISTANCE 2.  SAME AS DIST1 EXCEPT FOR Y AXIS.
C              XLENG,     DEFINES PHYSICAL LENGTH OF RESPECTIVE AXIS
C              YLENG      FOR VERSATEC PLOT.
```

```
C
C
C
C      STEPS IN PROCESSING ARE :
C      1.  INITIALIZE SYSTEM
C      2.  INPUT SAMPLE DATA
C      3.  IF DESIRED, OUTPUT SAMPLE DATA TO LINE PRINTER
C      4.  COMPUTE N SPACE INFORMATION CONTEXT RATIO
C      5.  READ IN NUMBER OF REFERENCE POINTS TO TEST
C      6.  UNTIL NO MORE REFERENCE POINTS TO PROCESS
C               READ A REFERENCE POINT
C               REDUCE SAMPLE DATA TO M SPACE USING THAT
C               REFERENCE POINT
C
C
C      PROCEDURE TO MODIFY CODE FOR VARIOUS SAMPLE FEATURE
C      CONFIGURATIONS
C      1.  MODIFY DIMENSION STATEMENTS TO THE VALUE OF THE INDICATED
C          USER SUPPLIED VALUES :
C             SAMPLE(NUMSAM,SAMSIZ)
C             UTILE1(NOCLAS,NOFEA)
C             UTILE2(NOCLAS)
C             UTILE3(NOCLAS)
C             UTILE4(NUMSAM)
```

```
C              UTILE5(NOCLAS)
C              UTILE6(NUMSAM,MSIZ)
C              REF2(NUMFEA)
C              DIST1(LGCLSZ)
C              DIST2(LGCLSZ)
C              ID(NUMSAM)
C              BUFFER(INPUT SAMPLE SIZE) *
C             *IF RAW DATA OF A VECTOR IS LARGER IN SIZE THAN NUMFEA
C              DESIRED TO  PROCESS FOR A REFERENCE POINT,
C              OTHERWISE USE SAMSIZ.  THIS IS INPUT DATA FORMAT
C              DEPENDENT
C         2.  MODIFY THE INITIALIZATION STATEMENTS AS APPROPRIATE.
C         3.  MODIFY THE GET DATA SECTION AS APPROPRIATE FOR THE INPUT
C             SAMPLE FORMAT.
C         4.  MODIFY THE REFERENCE POINT 2 INPUT SECTION TO THE APPROPRIATE
C             FORMAT STATEMENT FOR THE VECTOR SIZE.
C
C
      LOGICAL NOPRNT, NOPLOT
      INTEGER*4 NUMSAM,NUMFEA,SAMSIZ,UTILE1,UTILE2,UTILE3,UTILE4,UTILE5
     1 ,CLASS,ID,MSPACE,NOCLAS,NUMREF,UTILE6,MSIZ,LGCLSZ,SAMCNT
      REAL*4 SAMPLE, REF2,BUFFER
      REAL*4 DIST1,DIST2,XLENG,YLENG
      DIMENSION SAMPLE(0423,33),UTILE1(3,32),UTILE2(03),UTILE3(03),
```

```
     1 REF2(32),UTILE4(32),UTILE5(3),UTILE6(0423,3),DIST1(145),
     2 DIST2(145),ID(1080),BUFFER(63)
C
C       INITIALIZATION
C
        NUMSAM = 1080
        NUMFEA = 32
       SAMCNT = 0
        SAMSIZ = NUMFEA + 1
        NOCLAS = 3
        NOPRNT = .TRUE.
        NOPLOT = .FALSE.
        MSPACE = 2
        MSIZ   = MSPACE + 1
        LOCLSZ = 360
        XLENG  = 5.0
        YLENG  = 8.0
C
C       GET DATA
C
       DO 999  I = 1,NUMSAM
          READ(4,910) (BUFFER  (J),J=1,63),CLASS,ID(I)
 0910      FORMAT(7(8F10.7,/ ),7F10.7,2I5)
           IF (.NOT.(((CLASS.EQ.3).OR.(CLASS.EQ.4).OR.(CLASS.EQ.5)).AND.
```

```
     1    (ID(I).EQ.1)))  GO TO 0999
           SAMCNT = SAMCNT + 1
          DO 920  L = 1,NUMFEA
 0920        SAMPLE(SAMCNT,L) = AINT(((BUFFER  (L) + 8.0) * 32.0) + 0.5)
          SAMPLE(SAMCNT,SAMSIZ) = FLOAT(CLASS) - 2.0
           ID(SAMCNT) = ID(I)
 0999 CONTINUE
      NUMSAM = SAMCNT
      WRITE(6,1010) NUMSAM
 1010 FORMAT(1H0,'NUMSAM = ',I5)
      IF (NOPRNT) GO TO 1000
         WRITE(6,930) (SAMPLE(SAMCNT,J),J=1,SAMSIZ),ID(SAMCNT)
 0930      FORMAT(1X,4(8F10.2/,1X),    F10.2,I5/)
C
C      COMPUTE SCATTER WITHIN S(W) AND SCATTER BETWEEN S(B)
C
 1000  CALL SCATTR(SAMPLE,NUMSAM,NUMFEA,NOCLAS,SAMSIZ,UTILE1,LTILE2,
     1  UTILE3,LTILE4,UTILE5)
C    1
C     READ IN NUMBER OF REFERENCE POINT 2 VALUES TO PROCESS
C
       READ(5,2000) NUMREF
 2000  FORMAT(I3)
C
```

```
C        ITERATIVELY PROCESS REFERENCE VALUES
C
      DO 9999  I = 1,NUMREF
         READ(5,3000) (REF2(J),J=1,NUMFEA)
 3000    FORMAT(4(8F10.0,/))
C
C        TRANSFORM FROM N TO M SPACE
C
       WRITE(6,3005)
 3005 FORMAT(1H1,'UTILIZING REF PT 2 =',//,1X,'(')
      WRITE(6,3010) (REF2(M),M=1,NUMFEA)
 3010 FORMAT(1H ,8(F9.0,','))
      WRITE(6,3015)
 3015 FORMAT(1H+,82X,')')
       CALL REDUCE(SAMPLE,NUMSAM,SAMSIZ,NUMFEA,LTILE1,LTILE2,LTILE3,
    1  LTILE4,LTILE5,LTILE6,REF2,NOPRNT,NOCLAS,MSPACE,MSIZ,NOPLOT,
    2 LOCLSZ,DIST1,DIST2,XLENG,YLENG)
 9999 CONTINUE
 8888 STOP
     DEBUG SUBCHK
      END
```

```
C     SUBROUTINE SCATTR.                                                MAP00730
C      THIS MODULE COMPUTES THE SCATTER WITHIN AND SCATTER BETWEEN      MAP00740
C      CLASSES.  IT ALSO COMPUTES THE RATIO OF THE SCATTER WITHIN       MAP00750
C      S(W) AND THE SCATTER BETWEEN S(B).    UNEQUAL CLASS SIZES        MAP00760
C      ARE ALLOWED.                                                     MAP00770
C                                                                       MAP00780
C***********************************************************************MAP00790
C       OUTPUT OF THIS MODULE CONFIGURED FOR 32 FEATURES PER SAMPLE     MAP00800
C***********************************************************************MAP00810
C      INPUTS   : NUMBER OF SAMPLES                                     MAP00850
C       OUTPUT OF THIS MODULE CONFIGURED FOR 32 FEATURES PER SAMPLE     MAP00800
C                                                                       MAP00820
C                                                                       MAP00830
C                                                                       MAP00840
C                 SAMPLE DATA ITEMS                                     MAP00860
C                 NUMBER OF FEATURES PER SAMPLES                        MAP00870
C                 NUMBER OF CLASSES WITHIN THE SAMPLES                  MAP00880
C                 5 UTILITY ARRAYS CONTAINING DON'T CARE VALUES         MAP00890
C      OUTPUT   :  TO THE LINE PRINTER                                  MAP00900
C                    SCATTER WITHIN CLASSES                             MAP00910
C                    SCATTER BETWEEN CLASSES                            MAP00920
C                    RATIO OF S(W) / S(B)                               MAP00930
C                    MEANS FOR EACH CLASS                               MAP00940
C                    TOTAL SAMPLE MEAN                                  MAP00950
```

```
C                      NUMBER OF SAMPLES INPUT                                MAP00960
C                      NUMBER OF FEATURES PER SAMPLE                          MAP00970
C                      NUMBER OF CLASSES                                      MAP00980
C                   :  TO THE CALLING MODULE                                  MAP00990
C                      SAMPLE DATA ITEMS       UNCHANGED FROM INPUT           MAP01000
C                      NUMBER OF SAMPLES            "      "     "            MAP01010
C                      NUMBER OF FEATURES           "      "     "            MAP01020
C                      NUMBER OF CLASSES            "      "     "            MAP01030
C                      5 UTILITY ARRAYS   OUTPUT VALUE IS DON'T CARE          MAP01040
C                                                                             MAP01050
C     VARIABLES :                                                             MAP01060
C                                                                             MAP01070
C               SAMPLE   ARRAY OF DATA POINTS                                 MAP01080
C               NUMSAM   NUMBER OF SAMPLES                                    MAP01090
C               NUMFEA   NUMBER OF FEATURES, IE, THE NUMBER OF                MAP01100
C                        DIMENSIONS                                           MAP01110
C               SAMSIZ   NUMFEA + 1, THE NUMBER OF FEATURES PLUS 1            MAP01120
C                        ELEMENT CONTAINING THE CLASS ID NUMBER               MAP01130
C               NOCLAS   NUMBER OF CLASSES WITHIN THE SAMPLE SET,             MAP01140
C                        CLASSES MUST BE NUMBERED CONSECUTIVELY FROM 1        MAP01150
C                        TO NOCLAS. ZERO IMPLIES NO CLASS ASSIGNED            MAP01160
C               MEANCL   ARRAY OF THE SAMPLE MEAN FOR EACH CLASS              MAP01170
C               SCATWI   ARRAY OF SIZE NUMFEA.  CONTAINS THE SUM              MAP01180
C                        OF THE DIFFERENCES SQUARED BETWEEN EACH SAMPLE       MAP01190
```

```
C                       AND THE SAMPLE MEAN FOR THAT CLASS.             MAP01200
C             SCATBW    ARRAY OF SIZE NOCLAS, CONTAINS THE DIFFERENCES  MAP01210
C                       SQUARED BETWEEN TOTAL SAMPLE MEAN AND EACH      MAP01220
C                       CLASS MEAN.  EACH ARRAY ELEMENT IS THE SUM      MAP01230
C                       OVER THAT FEATURE.                              MAP01240
C             TOTMN     TOTAL MEAN,  THE MEAN DATA SAMPLE OVER ALL      MAP01250
C                       SAMPLES                                         MAP01260
C             CLASS     TEMP INDEX VARIABLE USED IN SUMMING MEANCL      MAP01270
C             CLCNT     CLASS COUNT, ARRAY OF COUNTERS TO DETERMINE     MAP01280
C                       THE NUMBER OF SAMPLES IN EACH CLASS             MAP01290
C             WITHIN    SUM OF WITHIN CLASS SCATTER OVER ALL CLASSES    MAP01300
C             BETWEN    SUM OF BETWEEN CLASS SCATTER OVER ALL CLASSES   MAP01310
C             RATIO     RATIO OF WITHIN / BETWEN                        MAP01320
C                                                                       MAP01330
C       STEPS IN PROCESSING ARE :                                       MAP01340
C       1.  INITIALIZE DATA                                             MAP01350
C       2.  DETERMINE THE MEAN SAMPLE FOR EACH CLASS                    MAP01360
C       3.  COMPUTE THE WITHIN CLASS SCATTER                            MAP01370
C       4.  DETERMINE THE TOTAL MEAN SAMPLE FOR THE ENTIRE DATA SET     MAP01380
C       5.  COMPUTE THE BETWEEN CLASSES SCATTER                         MAP01390
C       6.  COMPUTE THE RATIO OF SCATTER WITHIN DIVIDED BY SCATTER      MAP01400
C           BETWEEN CLASSES                                             MAP01410
C                                                                       MAP01420
C      PROCEDURE TO MODIFY CODE FOR VARIOUS SAMPLE FEATURE SIZES        MAP01430
```

```
C        1.  MODIFY FORMAT STATEMENT 1205 TO THE DESIRED NUMBER OF         MAP01440
C            FEATURES PER SAMPLE                                           MAP01450
C        2.  CHANGE CONFIGURATION STATEMENT IN PROGRAM COMMENTS ABOVE      MAP01460
C                                                                          MAP01470
      SUBROUTINE SCATTR (SAMPLE,NUMSAM,NUMFEA,NOCLAS,SAMSIZ,               MAP01480
     1 MEANCL,SCATWI,CLCNT,TOTMN,SCATBW)                                   MAP01490
C                                                                          MAP01500
      REAL*4 SAMPLE                                                        MAP01510
      INTEGER*4 NUMSAM,NUMFEA,NOCLAS,CLCNT,SAMSIZ,CLASS                    MAP01520
      REAL*4 MEANCL,SCATWI,SCATBW,WITHIN,TOTMN,RATIO,BETWEN                MAP01530
      DIMENSION SAMPLE(NUMSAM,SAMSIZ),MEANCL(NOCLAS,NUMFEA)                MAP01540
      DIMENSION SCATWI(NOCLAS),CLCNT(NOCLAS),TOTMN(NUMFEA)                 MAP01550
      DIMENSION SCATBW(NOCLAS)                                             MAP01560
C                                                                          MAP01570
C     INITIALIZE                                                           MAP01580
C                                                                          MAP01590
      DO 1000 I = 1, NOCLAS                                                MAP01600
        CLCNT(I) = 0                                                       MAP01610
        SCATWI(I) = 0.0                                                    MAP01620
        SCATBW(I) = 0.0                                                    MAP01630
        DO 1000 J = 1,NUMFEA                                               MAP01640
 1000     MEANCL(I,J) = 0.0                                                MAP01650
C                                                                          MAP01660
C     SUM EACH FEATURE OVER ITS CLASS                                      MAP01670
```

```
C                                                                      MAP01680
      DO 1110 J = 1, NUMSAM                                            MAP01690
         CLASS = IFIX(SAMPLE(J,SAMSIZ))                                MAP01700
         CLCNT(CLASS) = CLCNT(CLASS) + 1                               MAP01710
         DO 1100 K = 1 , NUMFEA                                        MAP01720
 1100       MEANCL(CLASS,K) = SAMPLE(J,K) + MEANCL(CLASS,K)            MAP01730
 1110  CONTINUE                                                        MAP01740
C                                                                      MAP01750
C     COMPUTE MEAN CLASS SAMPLE BY DIVIDING EACH FEATURE BY THE        MAP01760
C     NUMBER OF SAMPLES IN THAT CLASS                                  MAP01770
C                                                                      MAP01780
      DO 1210 I = 1, NOCLAS                                            MAP01790
         DO 1200 J = 1,NUMFEA                                          MAP01820
 1200       MEANCL(I,J) = MEANCL(I,J) / FLOAT(CLCNT(I))                MAP01830
 1210 CONTINUE                                                         MAP01860
      WRITE(6,1215)                                                    MAP01861
 1215 FORMAT(1H0,'MEAN SAMPLE FOR :   CLASS 1',10X,'CLASS 2',10X,      MAP01862
     1'CLASS 3')                                                       MAP01863
      DO 1225  J = 1,NUMFEA                                            MAP01864
          WRITE(6,1220) (MEANCL(I,J),I = 1,NOCLAS)                     MAP01865
 1220     FORMAT(1H ,14X,3(3X,F24.3))                                  MAP01866
 1225 CONTINUE                                                         MAP01867
C                                                                      MAP01870
C                                                                      MAP01880
```

```
C     COMPUTE WITHIN CLASS SCATTER WHERE SCATTER IS DIFFERENCE SQUARED  MAP01890
C     BETWEEN A SAMPLE AND ITS CLASS MEAN SAMPLE                        MAP01900
C                                                                       MAP01910
      DO 1399 I = 1, NUMSAM                                             MAP01920
        CLASS = IFIX(SAMPLE(I,SAMSIZ))                                  MAP01930
       DO 1300 J = 1, NUMFEA                                            MAP01940
 1300     SCATWI(CLASS) = SCATWI(CLASS) + (SAMPLE(I,J) -               MAP01950
     1                        MEANCL(CLASS,J)) ** 2                     MAP01960
 1399  CONTINUE                                                         MAP01970
      WITHIN = 0.0                                                      MAP01980
      DO 1410 I = 1, NOCLAS                                             MAP01990
       WITHIN = SCATWI(I) + WITHIN                                      MAP02000
       WRITE(6,1405) I,SCATWI(I)                                        MAP02010
 1405  FORMAT(1H0,'CLASS',I3,' SCATTER WITHIN = ',F24.6)                MAP02020
 1410 CONTINUE                                                          MAP02030
      WRITE(6,1415) WITHIN                                              MAP02040
 1415 FORMAT(1H0,'TOTAL SCATTER WITHIN = ',F24.6//)                     MAP02041
C                                                                       MAP02060
C     COMPUTE TOTAL MEAN SAMPLE                                         MAP02070
C                                                                       MAP02080
      DO 1500 I = 1,NUMFEA                                              MAP02090
 1500     TOTMN(I) = 0.0                                                MAP02100
      DO 1510 I = 1,NOCLAS                                              MAP02110
       DO 1505 J = 1,NUMFEA                                             MAP02120
```

```
 1505      TOTMN(J) = TOTMN(J) + (CLCNT(I) * MEANCL(I,J))                MAP02130
 1510 CONTINUE                                                           MAP02140
      DO 1520 I = 1,NUMFEA                                               MAP02150
         TOTMN(I) = TOTMN(I) / NUMSAM                                    MAP02160
 1520 WRITE(6,1525) I, TOTMN(I)                                          MAP02170
 1525 FORMAT(1H ,'TOTAL MEAN (',I4,') = ',F24.6)                         MAP02180
C                                                                        MAP02190
C     COMPUTE BETWEEN CLASS SCATTER S(B)                                 MAP02200
C                                                                        MAP02210
      DO 1605 I = 1,NOCLAS                                               MAP02220
         SCATBW(I) = 0.0                                                 MAP02230
       DO 1605 J = 1, NUMFEA                                             MAP02240
 1605      SCATBW(I) = SCATBW(I) + (MEANCL(I,J) - TOTMN(J)) ** 2         MAP02250
      BETWEN = 0.0                                                       MAP02260
      DO 1610 I = 1, NOCLAS                                              MAP02270
 1610    BETWEN = BETWEN + (SCATBW(I) * CLCNT(I))                        MAP02280
      RATIO = WITHIN / BETWEN                                            MAP02290
      WRITE(6,1700) NUMSAM,NUMFEA,NOCLAS                                 MAP02300
 1700 FORMAT(1H0,'FOR',I4,' SAMPLES, EACH WITH ',I4,' FEATURES, DIVIDED MAP02310
     1 INTO',I4,' CLASSES')                                              MAP02320
      WRITE(6,1705) WITHIN,BETWEN,RATIO                                  MAP02330
 1705 FORMAT(1H0,'THE SCATTER WITHIN CLASSES = ',F24.6/,1X,' THE SCATTER MAP0234
     1BETWEEN CLASSES = ',F24.6//' YIELDING A RATIO OF ',F12.6/////)     MAP02350
       RETURN                                                            MAP02360
```

```
DEBUG SUBCHK                            MAF02370
 END                                    MAF02380
```

```
C     SUBROUTINE REDUCE.  THIS MODULE TRANSFORMS SAMPLES IN N           MAP02390
C     DIMENSIONS ONTO M DIMENSIONAL SPACE, WHERE N >> M.                MAP02400
C     IT WILL OUTPUT THE M SPACE SAMPLE VECTORS TO THE LINE PRINTER,    MAP02410
C     IF SELECTED BY THE USER.                                          MAP02420
C                                                                       MAP02430
C      INPUT : SAMPLE DATA ITEMS                                        MAP02440
C             NUMBER OF SAMPLES IN DATA SET                             MAP02450
C             NUMBER OF FEATURES PER SAMPLE                             MAP02460
C             NUMBER OF ELEMENTS PER SAMPLE VECTOR                      MAP02470
C             PRINT OUTPUT INDICATOR                                    MAP02480
C             REFERENCE POINT 2 VECTOR                                  MAP02490
C             LOWER DIMENSIONAL SPACE DESIRED                           MAP02500
C             3 UTILITY ARRAYS                                          MAP02510
C     OUTPUT : TO LINE PRINTER                                          MAP02520
C                N SPACE SAMPLE NUMBER INDEX                            MAP02530
C                M SPACE DISTANCE VECTOR FOR EACH N SPACE SAMPLE INPUT MAP02540
C              TO CALLING PROGRAM                                       MAP02550
C                NUMBER OF ELEMENTS PER SAMPLE VECTOR UNCHANGED FROM    MAP02560
C                                                     INPUT             MAP02570
C                PRINT OUTPUT INDICATOR                 "        "      MAP02580
C                REFERENCE POINT 2 VECTOR               "        "      MAP02590
C                LOWER DIMENSIONAL SPACE DESIRED        "        "      MAP02600
C                3 UTILITY ARRAYS                     CON'T CARE        MAP02610
C                                                                       MAP02620
```

```
C                                                                       MAP02630
C     VARIABLES :                                                       MAP02640
C             SAMPLE  ARRAY OF DATA POINTS IN N DIMENSIONS              MAP02650
C             NUMSAM  NUMBER OF DATA POINTS IN  SAMPLE                  MAP02660
C             NUMFEA  NUMBER OF FEATURES PER SAMPLES                    MAP02670
C             SAMSIZ  NUMBER OF FEATURES + ONE, WHERE THE LAST          MAP02680
C                     ELEMENT IN THE VECTOR CONTAINS THE CLASS          MAP02690
C                     IDENTIFICATION NUMBER                             MAP02700
C             DIST1   FIRST FEATURE IN M SPACE, IT IS COMPUTED AS THE MAP02710
C                     EUCLIDEAN DISTANCE SQUARED FROM THE ORIGIN TO     MAP02720
C                     THE SAMPLE DATA POINT IN N DIMENSIONS             MAP02730
C             DIST2   SECOND FEATURE IN M SPACE, IT IS COMPUTED AS THEMAP02740
C                     EUCLIDEAN DISTANCE SQUARED FROM THE SECOND        MAP02750
C                     REFERENCE POINT TO THE SAMPLE DATA POINT IN N     MAP02760
C                     DIMENSIONS                                        MAP02770
C             DIST3   ARRAY OF CLASS LABELS RETAINED IN MAPPING SAMPLEMAP02780
C                     (I) FROM N SPACE TO M SPACE                       MAP02790
C               ***** NOTE  :  THE M DIMENSIONAL VECTOR COMPUTED FROM MAP02800
C                     A N SPACE VECTOR IS REPRESENTED BY THE ELEMENTS MAP02810
C                     OF THE THREE ARRAYS DIST1,DIST2,DIST3 SHARING     MAP02820
C                     THE SAME INDEX VALUE (I)                          MAP02830
C             REF2    VECTOR CONTAINING THE COORDINATES OF THE SECOND MAP02840
C                     REFERENCE POINT AS DETERMINED BY PROGRAM          MAP02850
C                     SEARCHR2                                          MAP02860
```

```
C           NOPRNT  LOGICAL VARIABLE : IF TRUE OUTPUT OF DIST1,DIST2MAP02870
C                                              DIST3 SUPPRESSED     MAP02880
C                                         IF FALSE DIST1,DIST2,DIST3    MAP02890
C                                              OUTPUT TO PRINTER    MAP02900
C           PTR     UTILITY ARRAY USED BY THE LIBRARY ROUTINE VSRTR MAP02910
C                   TO RETURN THE ORIGINAL ORDERING OF THE SAMPLE   MAP02920
C                   DATA IN SORTED ORDER BY DIST1                   MAP02930
C           MSPACE  NUMBER OF FEATURES IN LOWER DIMENSIONAL SPACE M MAP02940
C                                                                   MAP02950
C                                                                   MAP02951
C     STEPS IN PROCESSING ARE :                                     MAP02952
C     1.  COMPUTE 2 SPACE VECTOR FROM N SPACE VECTOR FOR EVERY SAMPLE. MAP02953
C     2.  CALL SCATTR TO COMPUTE S(W) AND S(B) IN 2 SPACE           MAP02954
C     3.  IF PLOTTING, SCALE DATA AND CALL PLOT                     MAP02955
C     4.  IF PRINTING, OUTPUT 2 SPACE VECTORS TO LINE PRINTER       MAP02956
C     5.  RETURN TO MAIN                                            MAP02957
C                                                                   MAP02958
C     PROCEDURE TO MODIFY CODE FOR VARIOUS SAMPLE CONFIGURATIONS    MAP02959
C     1.  CHANGE FORMAT STATEMENT 4200 IF MSPACE .NE. 2 TO AS       MAP02960
C         APPROPRIATE.                                              MAP02961
C                                                                   MAP02962
C                                                                   MAP02963
      SUBROUTINE REDUCE (SAMPLE,NUMSAM,SAMSIZ,NUMFEA,UTILE1,UTILE2,    MAP02970
     1 UTILE3,UTILE4,UTILE5,DIST,REF2,NOPRNT,NCLAS,MSPACE,MSIZ,NOPLOT, MAP02980
```

```
     2  LGCLSZ,DIST1,DIST2,XLENG,YLENG)                                  MAP02990
C                                                                        MAP03000
      REAL*4 SAMPLE                                                      MAP03010
      INTEGER*4 NUMSAM,NUMFEA,SAMSIZ,MSPACE,MSIZ,UTILE3                  MAP03020
      INTEGER*4 KEY,ISYM,ILINE,NP,LGCLSZ                                 MAP03030
      REAL*4 DIST,REF2,TMPMIN,TMPMAX,DIFMAX                              MAP03040
      REAL*4 UTILE1,UTILE2,       UTILE4,UTILE5,XLENG,YLENG              MAP03050
      REAL*4 XMAX,XMIN,YMAX,YMIN,DIST1,DIST2                             MAP03060
      LOGICAL NOPRNT,NOPLOT                                              MAP03070
      DIMENSION SAMPLE(NUMSAM,SAMSIZ),REF2(NUMFEA)                       MAP03080
      DIMENSION UTILE1(NOCLAS,MSPACE),UTILE2(NOCLAS),UTILE3(NOCLAS),     MAP03090
     1 UTILE4(MSPACE),UTILE5(NOCLAS)                                     MAP03100
       DIMENSION DIST(NUMSAM,MSIZ),DIST1(LGCLSZ),DIST2(LGCLSZ)           MAP03110
C                                                                        MAP03120
C                                                                        MAP03130
      DO 2100 I = 1,NUMSAM                                               MAP03140
        DO 2003 J = 1,MSIZ                                               MAP03150
 2003     DIST(I,J) = 0                                                  MAP03160
         DO 2005 J = 1,NUMFEA                                            MAP03170
            DIST(I,1) = DIST(I,1) + SAMPLE(I,J) ** 2                     MAP03180
            DIST(I,2) = DIST(I,2) + (SAMPLE(I,J) - REF2(J)) ** 2         MAP03190
 2005     CONTINUE                                                       MAP03200
          DIST(I,MSIZ) = SAMPLE(I,SAMSIZ)                                MAP03210
 2100  CONTINUE                                                          MAP03220
```

```
C                                                                       MAPO3230
C       COMPUTE S(W) AND S(B) FOR THIS ITERATION                        MAPO3240
C                                                                       MAPO3250
      CALL SCATTR(DIST,NUMSAM,NSPACE,NOCLAS,MSIZ,LTILE1,UTILE2,         MAPO3260
     1  UTILE3,LTILE4,UTILE5)                                           MAPO3270
C                                                                       MAPO3280
C       PLOT DATA IN N SPACE                                            MAPO3290
C                                                                       MAPO3300
        IF (NOPLOT) GO TO 4000                                          MAPO3310
C                                                                       MAPO3320
C       FIND MAX DIST1,DIST2 AND MIN DIST1                              MAPO3330
C                                                                       MAPO3340
          XMIN = DIST(1,1)                                              MAPO3350
          XMAX = DIST(1,1)                                              MAPO3360
          TMPMAX = DIST(1,2)                                            MAPO3370
          DO 3000  I = 2,NUMSAM                                         MAPO3380
            IF (DIST(I,1).GT. XMAX)  XMAX = DIST(I,1)                   MAPO3390
            IF (DIST(I,1).LT. XMIN)  XMIN = DIST(I,1)                   MAPO3400
            IF (DIST(I,2).GT. TMPMAX)  TMPMAX = DIST(I,2)               MAPO3410
 3000     CONTINUE                                                      MAPO3420
C                                                                       MAPO3430
C         SCALE DIST2 FOR PLOTTING                                      MAPO3440
C                                                                       MAPO3450
          DIFMAX = TMPMAX - DIST(1,2)                                   MAPO3460
```

```
      DO 3005  I = 1,NUMSAM                                             MAP03470
        DIST(I,2) = TMPMAX - DIST(I,2)                                  MAP03480
        IF (DIST(I,2) .GT. DIFMAX)   DIFMAX = DIST(I,2)                 MAP03490
 3005   CONTINUE                                                        MAP03500
      YMIN = 0.0                                                        MAP03510
      YMAX = DIFMAX                                                     MAP03520
C                                                                       MAP03530
C                                                                       MAP03540
C     INITIALIZE REMAINING PLOTTER PARAMETERS                           MAP03550
C                                                                       MAP03560
      K = 1                                                             MAP03570
      L = 0                                                             MAP03580
      KEY = 0                                                           MAP03590
      ISYM = 0                                                          MAP03600
      ILINE = 0                                                         MAP03610
      DO 3020  I = 1,NOCLAS                                             MAP03620
        KEY = KEY + 1                                                   MAP03630
        ISYM = ISYM + 1                                                 MAP03640
        M = 1                                                           MAP03650
        L = LTILE3(I) + L                                               MAP03660
        DO 3010  J = K,L                                                MAP03670
          DIST1(M) = DIST(J,1)                                          MAP03680
          DIST2(M) = DIST (J,2)                                         MAP03690
          M = M + 1                                                     MAP03700
```

```
3010       CONTINUE                                                      MAP03710
        CALL PLOTG(DIST1,DIST2,LTILE3(I),KEY,ILINE,ISYM,'DISTANCE**2 FMAP03720
    1ROM ORIGIN',23,'DISTANCE**2 FROM REF. PT. 2',27,XMIN ,XMAX,        MAP03730
    2   YMIN,YMAX,XLENG,YLENG)                                           MAP03740
3020    K = K + LTILE3(I)                                                MAP03750
        CALL PLOT(0.0,0.0,999)                                           MAP03760
4000  CONTINUE                                                           MAP03770
      IF (NOPRNT) GO TO 5000                                             MAP03780
      WRITE(6,4100)                                                      MAP03790
4100  FORMAT(1H ,6X,'DIST1',6X,'DIST2 SCALED',6X,'CLASS')                MAP03800
      WRITE(6,4200) (DIST(M,1),DIST(M,2),DIST(M,3),M=1,NUMSAM)           MAP03810
4200  FORMAT(1H ,2(6X,F12.0),15X,F2.0)                                   MAP03820
      WRITE(6,4300)  TMPMAX                                              MAP03830
4300  FORMAT(1H ,'MAX DIST2(UNSCALED) = ',F24.1)                         MAP03840
5000  CONTINUE                                                           MAP03850
      RETURN                                                             MAP03860
     DEBUG SUBCHK                                                        MAP03870
      END                                                                MAP03880
```

BIBLIOGRAPHY

[1] Duda, R. D. and Hart, P. E., Pattern Classification and Scene Analysis, Wiley-Interscience, 1973.

[2] Miesel, W. S., Computer-Oriented Approaches to Pattern Recognition, Academic Press, 1972.

[3] Andrews, H. C., Introduction to Mathematical Techniques In Pattern Recognition, Wiley-Interscience, 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52Bz Computer Science Department Naval Postgraduate School Monterey, California 93949	1
4.	Dr. Lonnie A. Wilson, Code 62Wi Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	10
5.	Dr. Richard Hamming, Code 52Hg Computer Science Department Naval Postgraduate School Monterey, California 93940	1
6.	LT. Michael L. Maurer, USN 14 Cliffside Drive Fort Worth, Texas 76134	3

Thesis 188677
M3847 Maurer
c.1
A dimensionality reduction technique for enhancing information context.